(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 483 664 B1

(12) EUROPEAN PATENT SPECIFICATION

(45) Date of publication and mention of the grant of the patent:
17.09.1997 Bulletin 1997/38

(51) Int Cl.[6]: **G06F 17/20**

(21) Application number: **91118111.3**

(22) Date of filing: **24.10.1991**

(54) Goal oriented electronic form system

Zielorientiertes elektronisches Formblattsystem

Système de formulaire électronique orienté vers un but

(84) Designated Contracting States:
AT BE CH DE DK ES FR GB GR IT LI LU NL SE

(30) Priority: **31.10.1990 US 606537**

(43) Date of publication of application:
06.05.1992 Bulletin 1992/19

(73) Proprietor: **BORLAND INTERNATIONAL, Inc.**
Scott's Valley, California 95066 (US)

(72) Inventor: **Turpin, William Monroe**
Santa Cruz, CA 95060 (US)

(74) Representative: **Blumbach, Kramer & Partner**
Patentanwälte,
Sonnenberger Strasse 100
65193 Wiesbaden (DE)

(56) References cited:
EP-A- 0 211 151 **US-A- 4 912 669**

- **THE COMPUTER JOURNAL. vol. 26, no. 1, February 1983, LONDON GB pages 52 - 59 N. H. GEHANI 'High Level Form Definition in Office Information Systems'**
- **5TH ANNUAL INTERNATIONAL CONFERENCE ON COMPUTERS AND COMMUNICATIONS 1986, pages 708 - 712 M. BUTTERWORTH 'Forms Definition Methods'**

EP 0 483 664 B1

Printed by Jouve, 75001 PARIS (FR)

CR

Description

This invention relates to a programmable controlled, goal oriented electronic system for form creation and form completion according to the preamble of claim 1.

Forms to gather data are employed daily in almost every commercial activity, in schools, and in all levels of government activity. It is a rare occurrence that an individuals life is not frequently touched by many forms. In the past, forms have been prepared by many processes ranging from hand and typewriter printed forms to engraved and mass produced forms. Prior to the advent of pervasive computing facilities, forms were completed by hand or by a typewriter and were generally interpreted by an individual. Today, there are many software packages which are capable of creating very fine printed forms. The recent proliferation of "Desk top publishing" software and of laser and inkjet printers has brought creation of good printed forms within the reach of individuals with high end personal computers as well as businesses.

Today, many electronic forms are completed by individuals using a keyboard and/or a mouse or other pointing device; the data thus gathered is possibly stored for later reference; and a report is printed for an immediate purpose. In prior art systems known to me, to the extent that forms provide prompting of fields to be completed, the fields are presented in sequence without regard for the data entered in the course of completing the form. If a form is extensive, there may be prompting for information which is not relevant in the context of the answers which have been entered. This is wasteful of operator time since unnecessary information is often requested.

In the prior art, in order to avoid prompting for unnecessary information, a first limited form is often presented for completion; the entries on that form are evaluated by an individual; and a decision is made to require completion of one or more additional forms. Since there is no automatic prompting for completion of additional forms which are dictated by answers on the completed form, the operator is unduly burdened with the decision process; and operator time is wasted.

Forms are often used to describe and organize a complex decision process or "business policy". As such, the form contains blanks for both the inputs and results of the decision process. However, the form itself it typically very poor at describing the decision process other than by including notes in the margins. For this reason, many forms are accompanied by an instruction sheet, or "policy manual", which the operator must read, interpret, and apply in the process of completing the form. This is wasteful of operator time, makes it harder to disseminate new decision processes, and results in many forms being completed incorrectly. This weakness of paper forms is not effectively addressed by current form software packages.

In "High Level Form Definition in Office Information Systems", The Computer Journal, Vol. 26, No. 1, pages 52-59, N.H. Gehani proposed a high level form definition language. The form definition language proposed is modeled after a procedural programming language. In "FORMS DEFINITION METHODS", 5th Annual International Conference on Computers and Communications 1986, pages 708-712 another form definition language is described by M. Butterworth. The form definition described is a procedural language in its use of subforms. However, both form definition languages are unintuitive, and therefore ill-suited for use by non-programmers. Form creation by means of such form definition languages need programmers, which are trained in procedural programming language, but is not suitable such as for office workers.

The invention, characterized in claim 1, is based on the problem to provide a goal oriented electronic system for form creation, form completion and using form data files which define:
(a) a graphical image of a goal oriented form for display on a monitor; and (b) a graphical image of at least one decision tree comprised of branches and conclusions which are discretely associated with fields of the form and which define logical and/or mathematical operations which implement goal oriented prompting within a form and among forms of a set of forms.

Further, in accordance with my invention, my system for generating form data files defines: (c) reading and writing links between fields of the form and a variety of data sources and destinations; and (d) other forms which, with the subject form, comprise a related set or "stack" of forms.

Provision of such inventive goal oriented electronic system which includes a graphical image of at least one decision tree defining these operations and provision of tools for modifying the graphical image of at least one decision tree ensure, that also non-programmers can create and complete interactive forms easily and quickly.

For purpose of clarification, a "goal oriented" electronic form is one in which the prompts for answers generally flow through the form from left to right; and from top to bottom of the form; and the ongoing pattern of prompting is conditioned on answers provided to the form or on data obtained from referenced sources. Advantageously, as the answers to the field prompts are entered, fields which need not be answered are skipped, and fields on the same or a linked form are prompted in the desired sequence.

In the event that an individual completing a report, by choice, revisits a completed field and enters a new value in the field, my form system automatically executes a prompting sequence consistent with that new value, and calculates new values for fields which are dependent on the value in the changed field. Advantageously, it is thus possible to try

various "what-if" scenarios. This feature of my system is termed "truth maintenance" since only valid and necessary prompting is implemented; and all calculated results are consistent with the values in the completed fields of a form.

In accordance with my invention, my system provides a set of intuitive "creation" tools which readily permit creation of the above referenced form files. In an illustrative embodiment of my invention, form creation is divided into four natural selectively reentrant activities: an initial specification of the fields of a form to be created; specification of the tree branches and conclusions to implement the intended logical and mathematical relations of the form; specification of reading and writing links to selected data files; and specification of relations between forms to define a stack of related interdependent forms.

Advantageously, these activities can be performed in any desired order; and each activity can be reentered selectively to make additions and/or corrections in order to accommodate thoughts which occur in the course of form creation.

Furthermore, at any point in the process of form file creation, it is possible to selectively display: the current form; any selected part or all of the related tree structure; links to data sources and destinations; and the contents of a stack and the order of the contents in the stack.

In accordance with my invention, if during the course of creating a form, an expression assigned to a branch or conclusion references a form field which does not exist, my system automatically creates a new field which adopts the established name. Subsequently, a field may be placed on the form to hold that name; however, if no field is assigned on the form, my system automatically prompts for a value at the appropriate place during the completion of the form. The prompt for such a field presents a prompt window that requests selection of a value for the question that does not appear on the form; however, a value is required for that field since continued prompting in the form is dependent on the value selected.

In accordance with my invention, if during the course of creating a form, links are requested to a data base which does not exist, my system automatically creates a new data base with fields, which adopt the established names and characteristics of the fields contained in the form system.

In accordance with my invention, "help" information may be assigned to a field during form creation; and that help information is available to an operator during form completion.

In accordance with my invention, I provide "run time" software for operator completion, but not alteration, of previously created forms. My "run time" software permits an operator to selectively view the trees associated with a form being completed to provide an understanding of the logical and mathematical relations and processes embodied in the form. Advantageously, my graphical tree displays identify "active" and "inactive" tree branches in accordance with data gathered in the form prior to display of the tree.

Advantageously, my form system automatically reformats horizontal segments of a graphical display of a tree that covers two or more horizontal segments and two or more vertical screens in order to minimize the number of vertical screen displays required to show the entire horizontal segment.

Advantageously, my system may be used to both create and complete goal oriented forms to implement inquiries in any situation in which the relations and functions of the fields of a form can be described by a tree of branches and conclusions.

Although my forms provide goal oriented prompting, an operator may choose to depart from the suggested order of form completion. In accordance with my invention I provide a "resume" function which may be manually selected to return to goal oriented prompting for further answers required to complete a form.

During completion of a form, a field may require selection of a value from a defined set of values. The list of values, from which a selection is to be made, may be created manually during form creation; or may be derived from tree statements which: (a) are attached to the field and create answers which correspond to the selections in the list; (b) rely upon selection of a value from the list to complete evaluation of an expression; or (c) are established by a link to a database which provides values contained therein.

In the course of form creation, the display of fields which require selection of a value from a set of values, as a design choice, may be defined as "selection list" fields or "check box" fields.

In the case of a "selection list" field, a dialog window with a list of values is presented for selection of a value when the corresponding field is prompted for an answer. A selection is made by moving a cursor over the desired item and clicking the mouse or depressing the return key.

In the case of a "check box" field, each value of the list is displayed with a small box for placing a check mark. In accordance with my invention, my form system automatically generates a field object which contains a number of selection boxes equal to the number of possible selections. Advantageously, my system automatically arranges the display of the set of selection boxes to match the size and shape of the field on the form. If the allotted field space is too small to accommodate all of the check boxes and their name text, the field is automatically defaulted to a "selection list" field.

In accordance with my invention, keyboard entries are checked against "field characteristics" which are assigned to a field during form creation. If a keyboard entry for a field is not consistent with the assigned characteristic, the

entered value is rejected and an error message advises the operator of a problem. Such characteristics can be assigned to a field by standard "picture" specifications. Alternatively, requirements for the form of a field input can be established by local form rules which are implemented by decision trees attached to the field. As an option, upon the occurrence of an error in input format, the field in error can be cleared and the prompt returned to that field to continue form completion.

In accordance with an aspect of my invention fields of a form may be designated as "protected" or "unprotected" at the time a form is created. Values cannot be entered manually in a "protected" field since only the values calculated for the field are considered valid. Even though a value may be automatically calculated for an "unprotected" field, a value may be entered into the field manually to handle exceptional conditions. Fields with this characteristic are termed "over ride" fields. Advantageously, in accordance with my invention, my system clearly marks or flags both the display and printing of fields which contain over ride values.

Fig. 1 is a block diagram of a personal computer
Fig. 2 is an overview of software employed in the personal computer of Fig. 1;
Fig. 3 is a general view of the major elements of my goal oriented form software;
Fig. 4 is a general view of a form image data file;
Fig. 5 illustrates an opening window of my form system application program and the menu commands available;
Fig. 6 illustrates a Form Tool window and the menu commands available;
Fig. 7 illustrates a Tree Tool window and the menu commands available;
Fig. 8 illustrates a Stack Tool window and the menu commands available;
Fig. 9 is the first form in a set of four forms for an application for life insurance example;
Fig. 10 is the four forms for an application for life insurance example;
Fig. 11 is the third form in a set of four forms for an application for life insurance example;
Fig. 12 is the fourth form in a set of four forms for an application for life insurance example;
Fig. 13 illustrates a window with a "goal" life insurance application for completion or modification;
Fig. 14 illustrates the display of a second form for prompting of values necessary for completion of a goal form;
Fig. 15 illustrates the highlighting of the selected path in a tree;
Fig. 16 illustrates the indication that a value for a field on a form has been overridden by a user;
Fig. 17 is the dialog box for attaching context sensitive help to a field;
Fig. 18 illustrates the automatic arrangement of check boxes in a vertical region;
Fig. 19 illustrates the automatic arrangement of check boxes in a horizontal region;
Fig. 20 illustrates the automatic presentation of a selection list when insufficient space is provided in a region for check boxes;
Fig. 21 is a dialog box for automatically or non-automatically specifying values expected for a field;
Fig. 22 is a dialog box for specifying field protection;
Fig. 23 illustrates a stack tool window with a display of related forms;
Fig. 24 is a display of a branch object in a tree;
Fig. 25 is a display of a conclusion object in a tree;
Fig. 26 illustrates multiple branches and expressions for calculating results for each branch;
Fig. 27 is a dialog box for specifying conditions and conclusions in a tree;
Fig. 28 is a dialog box for pasting functions into an expression;
Fig. 29 is a dialog box for pasting field names into an expression;
Fig. 30 illustrates a larger perspective view of a tree shown in Fig. 31.
Fig. 31 illustrates a more detailed view of a portion of the tree in Fig. 30.
Fig. 32 illustrates a self-referencing tree;
Fig. 33 is a dialog box for establishing links between fields in the form system and fields in related database(s);
Fig. 34 is a dialog box for selecting the option to create a new database file when there is no established file.

DETAILED DESCRIPTION

The illustrative embodiment of my invention is disclosed as an application program running under Microsoft WINDOWS™ graphical environment program on an IBM compatible PC.

Notwithstanding, disclosure of my invention in this particular environment, the principles of my invention can be implemented as a program which includes an integral interface facility; or in the context of other interface environments.

Although the graphical images and protocols employed by my form system are generally driven by the WINDOWS environment, my system includes menu features which are not present in or contemplated by WINDOWS. The general features, functions and protocol of WINDOWS are described later herein with the introduction of the opening window of Fig. 5.

Fig. 1 is a very general block diagram of an IBM compatible personal computer (PC) which supports the Microsoft WINDOWS graphical environment, and, in turn, WINDOWS supports my form system which is described herein.

The central processing unit (CPU) 100 typically employs a processor of the Intel™ family of microprocessors. The read only memory (ROM) 101 contains the basic input output system code (BIOS) for addressing and controlling floppy disk 103, hard disk 104 and printer 108. Random access memory (RAM) 102 is the working memory for CPU 100. In a typical WINDOWS installation, RAM of two megabytes or more is employed.

Monitor 105 of Fig. 1 provides a visual display; keyboard (KB) 106 and mouse 107 provide for manual input to any process running on the PC. Printer 108 creates hard copy images of output of the PC; and modem 109 provides communication between the PC of Fig. 1 and other computers.

In Fig. 1, hard disk 104 is illustrated as containing a body of program and data information 121. Included in this body of information is a disk operating system (DOS), the WINDOWS graphical environment system software; user application programs which operate under the WINDOWS environment; user application programs which do not employ the WINDOWS environment facilities; and data files of all sorts.

Fig. 2 illustrates, in a general way, the interaction and flow of information between the illustrated software entities.

Non-WINDOWS application programs 201-1 through 201-M are served by the CPU 100 operating under Microsoft Corporation MS DOS system 206. Programs and data flow between Non-WINDOWS applications 201-1 through 201-M and MS DOS 206 via paths labeled e.g., 210, 211. Paths 210, 211 are symbolic paths and are not intended to represent physical paths.

The MS DOS operating system 206 employs MS DOS software device drivers to control the disks 221 and printers 222 through the facilities of ROM BIOS 209. MS DOS device drivers also control system communication with the display monitor, an RS232 port, a keyboard, a modem and a mouse.

WINDOWS application programs 203-1 through 203-N are served by WINDOWS graphical environment software 205. The windows software comprises: User, graphical device interface (GDI) and Kernel modules. Symbolic communication paths 212 and 213 pass function calls to WINDOWS software 205 and responses to the respective WINDOWS application software.

WINDOWS device drivers 208 are the counterpart of MS DOS device drivers 207 and serve the same functions.

In Fig. 3, box 300 represents the major software modules of my form system. In accordance with my invention my form system comprises two modes of operation, namely "form creation" and "run time" form completion. Form creation comprises four phases:

(1) Definition of: a form image for all forms of an application, names of fields of the form or forms, and field characteristics;

(2) Definition of the forms of a related set i.e., a "stack" of forms and the assigned order of the forms in the set. When a form set is opened for completion, the defined order establishes which form of a set is the initial "goal" form, and the order in which the other forms of the set are presented for completion;

(3) Definition of decision tree structures comprising branches and conclusions which are assigned to the fields of the forms which comprise a related stack of forms; and

(4) Definition of reading and writing links between fields of a form and extrinsic data sources and destinations.

The four tool modules, 301 through 304 serve in the implementation of phases 1 through 4 referenced above herein. Tool modules 301 through 304 are not available in my run time form completion mode of operation.

Memory manager module 305 manages the assignment of memory space. This module performs common functions for the other modules relating to the allocation and deallocation of portions of memory to contain data structures. It does this by allocating large portions of memory from Windows and dividing these into smaller portions as needed by the other modules. The memory manager also maintains a list of names used for forms, fields, system functions, and links (called a symbol table) so that the portion of memory associated with these items can be located and referenced by its name.

Form execution module 306 and tree execution module 307 serve in implementation of my goal oriented form completion mode of operation. These modules are also available for use in conjunction with tools 301 through 304 during form creation.

Link manager module 308 implements reading and writing communication with the extrinsic data sources and destinations defined during form creation.

File I-O subsystem module 309, among other functions, controls the transfer and the form of data as it is moved between the hard disk and the RAM main memory of the PC.

WINDOWS interface module 310 provides communication between my form system and the WINDOWS graphical environment software.

Fig. 4 represents the major divisions of my "form image data file" which is generated during form creation and is maintained in disk memory. A detailed description of the "form image data file" of Fig. 4 is included herein as Appendix

A which appears immediately before the Claims.

File I-O Subsystem module 309 transfers a form image data file between main memory and the hard disk for storage and retrieval in the course of creation and completion of the form defined by the file. The image file stored in main memory and the corresponding image file stored in a hard disk contain the same data; however, the file in main memory is a binary representation of the image data, and the file in hard disk is an ASCII representation of the numerical and text portions of the image data. File I-O Subsystem module 309 makes the conversions during transfer of an image file.

At the time that a form image data file is transferred to main memory for editing or completion, my form system analyzes the data therein and constructs a symbol table, a set of memory structures which correspond to each record in the data file (forms, form objects, fields, tree objects, and links), and "linked lists" which represent dependencies between the various form system components. The symbol table is a list of all names used in the form and the memory location of the records of that list.

The linked list is required to determine the proper order for goal oriented prompting through the collection of forms. The linked list represents the data dependencies which are inherent in the decision tree definitions contained within the data file. These dependencies must be comprehended by the tree execution module when performing calculations or when determining the next field value to prompt for.

Three types of dependencies must be maintained for proper execution by the tree evaluation module:

(1) The use of a field as a branch condition within a decision tree. The value of the field must be determined before a branch can be selected.

(2) The use of a field within a formula that specifies the condition under which a branch should be taken. The value of the field must be determined before the condition can be evaluated.

(3) The use of a field within a formula that specifies the conclusion value at a terminal branch of a decision tree. The value of the field must be determined before the conclusion can be evaluated.

All three types of dependencies are constantly maintained in memory using linked lists and are updated as required when additions or modifications are made to decision trees via the tree tool module.

Figs. 5 through 8 illustrate various window presentations and pull down menu commands which may be encountered in the use of my form system.

Fig. 5 is an opening window which is displayed prior to selection of a form application. The menu items shown in the main body of Fig. 5 are displayed on a mutually exclusive basis when the corresponding menu items, File, Edit, etc. are selected. Since this is the first window described herein, the features which are derived directly from the Microsoft WINDOWS environment are provided as background to the later description of my form system.

In the terms of WINDOWS, software, such as my form system software, is called an application program. The term application as used in WINDOWS must be distinguished from forms by which an individual makes an "application" e. g., for credit approval. With the WINDOWS definition of the term "application" in mind, the WINDOWS environment provides for two general types of windows, namely, "application" windows which contain currently running application software and "document" windows which appear with application software that can display two or more windows simultaneously.

Document windows share the application window's menu bar. Commands that affect an application window affect the document as well. Document windows have their own title bar unless their physical size is maximized to fill the screen. In the latter case the document window and the application window share a title bar.

Fig. 5 illustrates the opening window of my form system application program. The small rectangle in the upper left corner of the window of Fig. 5 represents the window control menu box which is found on all windows of the WINDOWS environment. The pull down menu for the control menu box of Fig. 5 is shown under that heading in the working area of the window. The menu for the control menu box and the main menu items are shown for purposes of discussion only. These menus are displayed only after a main menu command has been selected.

The control menu commands permit an individual to: size, move, maximize, minimize and close windows; and to switch to WINDOWS Task List from a keyboard or by use of a mouse.

The horizontal area to the right of the control menu box in Fig. 5 is the title bar which designates an application program e.g., Form System as shown in Fig. 5; and the title of the current active files under the named application program. The down and up arrows on the right side of the title bar are employed respectively to decrease and increase the size of the window.

The pull down menu commands for the opening window, as described below herein, are tailored to my form system. When a pull down menu is displayed, the commands which are then available for execution are presented in a bold black type style; and the commands which are not available for execution are displayed in a readable, but somewhat obscured print style.

For purposes of complete understanding, all of the menu commands of Figures 5 through 8 are described in

Appendix B attached hereto.

Modes of operation

As indicated earlier herein, my form system has two modes of operation, namely, form creation and run time form completion. In the following discussion, a description of form creation follows a description of run time form completion. This order of presentation is adopted because the description of a previously created form provides valuable insights into my goal oriented forms, and to the decision trees, links and form stack relations embodied therein.

Form Completion

For purposes of illustration, a set of four forms for making application for life insurance are displayed in Figs. 9 through 12. The data file containing the life insurance forms is entitled Life.DF.

When form completion proceeds during a "run time" session of my form system, a subset of menu commands are available to the user. For example, none of the Tools (Forms, Tree, Stack and Link) are available.

In Fig. 5, an operator selects the "Open" command from the "File" menu. In response to this command, my form system provides a list of form files, including Life.DF, which are available for selection. A selection is made by highlighting the file to be selected and either clicking the mouse or striking the RETURN (or ENTER) key on the keyboard. Following selection of a form file e.g., Life.DF a screen essentially as shown in Fig. 13 is presented to an operator for completion. The form shown in the window of Fig. 13 is also shown in Fig. 9.

When a goal form e.g., the Life Insurance Application form is presented as shown in Fig. 13, the first field to be completed, Proposed Insured is outlined in a heavy line and a large "I" shaped cursor is presented in that field. Information input to a prompted field may comprise: typed information followed by depression of the RETURN key of the keyboard; or may comprise selection by use of a mouse or by use of the ARROW and RETURN keys of the keyboard.

In order to implement goal oriented prompting, my system first determines which form is the goal form. When an application is initially loaded into memory, the top form of the stack is selected as the goal form. Later, an operator can use the "Select" command on the "Form" menu to select another form to become the goal form.

Once a goal form has been selected, my form system selects the first field without a value on that form as the goal field. It does this by searching down the linked list of field objects on the form until it finds a field that does not currently have a value.

Once a goal field has been selected, my system next determines which field, if any, is dependent on the goal field. This is done by looking at any decision trees which are associated with the field to determine which field in the decision tree is next needed to complete the tree. This is done by starting at the base of the tree and following all selected branches of the tree until my system detects either a branch node that does not have a value, a condition expression that does not have a value, or a conclusion expression that does not have a value. This field, if any, becomes the dependent field which my form system must prompt for next.

Once my system has determined which field to prompt for, the system next locates any form that contains this field. Starting at the top of the stack, my form system looks at each form in turn to find which form closest to the top of the stack contains that field. My form system then moves that form to the top of the stack so that the user can enter a value. If the field is not found on any form, my system prompts for the field on a special "scratchpad" form.

Once the form containing the dependent field has been moved to the top of the stack, my system then positions the cursor on the dependent field and prompts the operator to enter a value for that field.

In the Life Insurance Application example shown in Figures 9 through 12, my system automatically prompts for fields contained on the Premium Calculation, Declarations, and Medical Information forms, as necessary, to complete the Life Insurance application form. Fig. 14 shows the display after the Premium Calculation form has been automatically moved to the top of the stack to prompt for "Amount of basic policy". This was done because my system determined that "Amount of basic policy" was the next dependent field necessary to calculate a value for the "Total Annual Premium" field on the Life Insurance Application form, which was the goal form. Since the Premium Calculation form was moved to the top of the stack temporarily due to my goal oriented prompting, it is identified as as prompt form by displaying the word "prompt" after the title of the form as shown in Fig. 14. This form will also be automatically removed from the display once the operator enters values for the dependent fields on it.

Rather than provide values for dependent fields, an operator can use the "Close" command on a prompt form's control menu to close the form at any time. When the operator does this, my system moves to the next field on the current goal form and proceeds with the goal oriented prompting for its dependent fields, if any.

An operator can also cause my system to pursue goal oriented prompting for any field of his or her choice by first selecting the field, then using the "Calculate" command on the "Field" menu. This causes my system to make the selected field the goal field for purposes of goal oriented prompting.

After a user has entered a value for a field, whether or not a prompted fild, my system must propagate that value

throughout any forms and decision trees associated with that field. I call this feature of my system "truth maintenance" because it maintains at all times the logical and/or mathematical relationships between fields on forms. The actual implementation of truth maintenance is based on the linked list structures that are created as a form image data file is transferred to main memory. The first step of this process is to remove the previous value, if any, of the field before the user typed a new entry. Once the previous value has been removed from the field, this change is propagated to any fields which are dependent upon that field to remove all prior dependent values. The second step is to place the newly entered value into the field; and to propagate the changes to all dependent fields.

My system then looks and determines which forms, if any, contain the field and displays the new value on each of those forms. If the goal form, which the system selected in its goal oriented prompting, now has a value for the field which was originally the goal field, or if the operator did not enter a value for the prompted field but rather answered a value for a different field, or if the operator pressed the Tab Key, then the goal form is advanced to the next field and the goal oriented prompting sequence starts over again for that field.

My form system also maintains any dependencies related to external sources of data that have been linked to the forms. When the value of a field that is used as an index for a database is modified, my system automatically locates the appropriate record and updates the values of any fields linked to the database. Similarly, when the value of a field that is exported to another application is modified, my system automatically notifies the other application of the change.

In the Life Insurance Application example shown in Fig. 13, when the operator enters the applicant's name, my system automatically looks in a database file for information about the applicant. If information about the applicant is found in the database file, the applicant's address, date of birth, etc. is retrieved from the file and the system automatically skips over these fields. If no information about the applicant is found in the database file, the system prompts the operator for this information.

Upon entry of a value for any field, my system automatically prompts for entry into the next field according to the goal sequence defined above. As values are entered into the prompted fields, automatic prompting may continue on the initial goal form to completion of that form; or dependent on the values entered in certain fields, prompting may digress to a subsidiary form of the stack. In any event, form fields which receive their data from linked data sources or by calculation are not visited by the prompting cursor.

If the cursor is manually moved to a field which receives data from a linked source or by calculation, the outline of the field is a distinctive dotted border to advise that the operator is not expected to provide an answer. In the illustrative Life Insurance Application form of Fig. 13, the fields: "Proposed insured", "Beneficiary name", "Beneficiary address", etc. are all fields for which the operator is prompted for an answer. On the other hand, the fields: "Total Annual premium", "Premium payment amount"; and "Deposit required" are fields which receive their values by calculations.

Fig. 15 illustrates the ability of the system of my invention to highlight the selected path in a tree for the user. In this case, the tree for "premium payment amount" is currently determined by the value first for the insured not meeting the basic requirements being "no" and the mode of payment being "monthly" with a thicker line for that selected path and then the calculation corresponding to monthly mode of payment is the expression which is used to calculate the premium payment amount.

Also of note in Fig. 15 is the use of different icons in the decision tree display to distinguish calculated fields. The leftmost branch object includes a decision tree icon above the branch field; in this case "Insured does not meet basic". This decision tree icon indicates that the value of "Insured does not meet basic" is calculated via a decision tree rather than being entered by the operator. The other branch object, for "Mode of payment", does not have this icon. "Mode of payment" is a field which the operator must enter. This display technique highlights the capability of my invention to embed arbitrarily complex computations which result in a value for a field within a single branch object.

Finally, in Fig. 16 is the capability of my invention to indicate that a value for a field has been entered by the user overriding the value that would be brought to that field from the tree. In this example, the field called "Premium Payment Amount" has been entered as $150.00 by the operator and the cross/hatching over that field indicates that this value was entered by the operator rather than by the tree that is available for that correct determination of the premium payment amount.

Form Creation

I contemplate that my form system will be widely used to create sets of forms for all types of commercial, industrial and other applications of my form system almost without limitation.

Form creation in my invention involves the use of four interrelated tools. The form tool, the stack tool, the tree tool, and the link tool. These will be discussed individually in the following paragraphs.

Form Tool

The form tool of my system is a facility for creating and modifying application forms. The form tool provides a high

level, graphical method for defining forms. It operates much like a drawing package and displays forms as they are being defined.

I view a form as a physical area which can be divided into a plurality of regions. The physical size of a region can be selectively set; and a region can have a border on any or all sides. The width of a region must be an integral multiple of the pitch of the default font employed in a form; and the height of a region must be an integral multiple of the height of the default font. The borders for adjacent regions are shared.

Form objects fall into two general classes, namely, static and dynamic. Regions which are assigned static form objects are not involved in my goal oriented prompting. The static form objects are: text, graphics, filled rectangles, rounded rectangles and lines. There is a single dynamic form object field. Each Field must have a name for identification and reference in trees, conclusions, and links.

There are three static form object regions in the illustrative insurance application of Fig. 13. The large title region with the text "Apex Life Insurance Company" and the signature region at the bottom of the form of Fig. 13 are both text form objects. The title region illustrates the use of text font type and size which are different from the default text. The region to the right of the region named "Premium payment amount" is a filled rectangle form object.

The remaining regions of the form of Fig. 13 are field form objects which are for ease of reference termed "fields" herein. Fields are employed to display: (a) data entered by a user; (b) data calculated by my form system; or (c) data provided by a link to an external source.

All form objects have assigned "properties" which define: size, appearance, and functions attributable to an object. For example, all form objects may be assigned a border property; and this is the only property which can be assigned to filled rectangle or graphics objects. Font and alignment properties, also, can be assigned to text objects.

In contrast to the limited number of properties available for assignment to the "static" form objects, a wide range of properties can be assigned to "fields". The properties which are available for assignment to field are enumerated in Fig. 5 under the menu heading "property". A description of these properties is to be found in Appendix B hereof, under the heading Properties.

Once a general plan for the forms of an application has been conceived, form creation begins with use of the Form Tool of my system. The operator invokes the Form tool by using the "Form" command on the "Tools" menu shown in Fig. 5.

The form tool provides the following capabilities: (a) creation of a new form; (b) adding new objects to a form; (c) renaming, sizing and scrolling forms; (d) finding forms that contain a specified field; (e) selecting, moving and sizing form objects; (f) editing form objects with the clipboard; (g) changing the field referenced by a field object; (h) changing the names of field and text objects; (i) adding help text to be displayed for a field object; (j) changing the display format of a field object; (k) changing the alignment of text within field objects and text objects; (1) changing the character fonts of text objects and field objects; (m) controlling which, if any, borders are drawn around objects; (n) controlling whether the field name is displayed in a field object; and (o) protecting field objects both from override by the operator or display of the tree associated with the field object.

The Life Insurance Application referenced earlier herein, as an example, illustrates several features which are provided by my form tool. Fig. 17 shows the dialog box for attaching context sensitive help to a field using the "Help" command on the "Properties" menu in Fig. 6. In this example, the help for the field called "Proposed Insured" is an elaboration of some information that may be of value to the operator filling out the form.

Fig. 18 and 19 illustrate an automatic feature provided in the form tool that places check boxes within the space allotted to a field. In Fig. 18 a vertical space is alloted a field called "Mode of Payment" and the check boxes are displayed accordingly. In Fig. 19 a horizontal field is provided for mode of payment and the check boxes are arranged accordingly. Fig. 20 shows the case where insufficient space is allocated for "mode of payment" and although check boxes are indicated, the system automatically provides a selection list since there is insufficient room for the check boxes. There is always room for the selection list since even as the list grows, scroll bars can be added to the display and an arbitrarily long list can be shown.

Fig. 21 shows a dialog box that allows for the automatic generation of the values for fields. This dialog box appears whenever the operator changes the type of a field to either "selection list" or "check box" using the "Field Type" command on the "Properties" menu shown in Fig. 6. The automatic determination of the values looks at values that can be attached from the tree, values that are used in a tree which employs the field for determination of the other tree's value, or finally automatic creation of the values by looking at the values that can be brought from the records of a database. If automatic is not selected, then the new values are manually entered in the edit box under "New Value" and then added to the list in the box called "Values".

Another capability of my invention is to provide protection of fields contained on forms and there are two different protection modes possible. Fig. 22 shows the dialog box that can be used to disallow override values using the "Protection" command on the "Properties" menu shown in Fig. 6. The meaning of no override is that the user is not allowed to override a value which has been assigned to the field from a tree or from a database reference. Field protection can also block the ability for the user of the application to observe the decision tree logic for a particular field. Both of these

protections are done on a field-by-field basis.

Stack Tool

The Stack Tool, which provides for manipulation of the forms of an application, is a high-level, graphical facility for copying, creating, deleting and arranging forms. Within the stack tool there are specific capabilities that allow application creators to create new forms, change the title of an existing form or change the order of the existing forms within an application. For instance, it is often useful to change the order of forms to move a new form to the top of the stack so that it becomes the goal form when the application is initially loaded into memory.

The stack for the Life Insurance Application used in the previous description of form completion is depicted in Fig. 23. Fig. 23 depicts a window which is displayed when the stack tool is chosen using the "Stack" command on the "Tools" menu. It shows the four related forms that comprise the "stack" or set of forms for this application. As seen in Fig. 23, the stack for the file Life.DF comprises the goal form and three subsidiary related forms. Of special note in Fig. 23 is the fact that the top form of the stack, in this case the Life Insurance Application form, is depicted as a goal form through the use of a special icon for the top-most form in the stack.

Tree Tool

In my invention, another specialized tool called the Tree Tool is provided in order to create and modify decision trees. The Tree Tool is invoked by the operator by first selecting the field associated with the tree and then using the "Tree" command on the "Tools" menu as shown in Fig. 5 and Fig. 6.

Two basic types of objects can be created using the tree tool. The first object is the branch object which is shown in Fig. 24 highlighted with a broken line. The branch object consists of a condition of the preceding field; in this case, Field 1. The first condition of Field 1, condition 1A, is the condition leading to the highlighted object. The second part of the branch object is the field upon which the new branch will be taken; in this case, Field 2.

Fig. 25 illustrates the conclusion object. The conclusion object is highlighted with a broken line. The conclusion object consists of a condition that the preceding field, again in this case Field 1. The second condition of Field 1, condition 1B, is the condition of this object. The second part of the conclusion object is the conclusion itself; in this case, just indicated with the word "Conclusion". Conclusions can be text, fields, functions, or combinations of the proceeding in expressions connected with operators using spreadsheet syntax.

Fig. 26 shows multiple branches from an example field called "Mode of Payment". If mode of payment is "annual", the value for the premium payment amount is the "total annual premium" as indicated in the conclusion for that branch. If the payments are made "semi-annually", the expression uses the function @ROUND of the total annual premium multiplied times the factor that it adjusts it for the fact that there are two payments made during the year (each equal to about one-half or 0.515 of the annual amount). The @ROUND function also requires specification of the number of decimal places. In this example, the value set at two places gives a dollar and cents amount. My system provides a complete set of built-in functions, such as @ROUND, which can be used within tree conditions and conclusions to calculate values based on field values. These functions are listed in Appendix A under the heading "IDFunction".

A dialog box like that shown in Fig. 27 is displayed as a part of the specification of both conditions and conclusions. This dialog box appears when the operator selects either the "Condition" or "Conclusion" command on the "Properties" menu shown in Fig. 7. The condition or conclusion expression is contained within the edit window in the upper part of the box. There are options to assist the entry process by providing pasting of functions and fields into the condition. For the case of pasting functions, Fig. 28 shows a portion of the list of functions available in alphabetical order including an option to paste in descriptive arguments for the functions. Fig. 29 shows the dialog box allowing the pasting of fields. This is simply a listing of all of the fields currently defined in the application thereby saving a number of keystrokes for the choice of a field from the list of all possible fields available.

My invention also provides a very innovative approach to the display of arbitrarily large trees in a fixed-size region, such as on a computer display. Figures 30 and 31 both display the same decision tree but at two different levels of magnification. Fig. 30 shows a larger view than that shown in Fig. 31. In Fig. 31 the fields, the branches, the conclusions are arranged with spacing to maximize the amount of information displayed. If a more magnified view is selected, like that of Fig. 31, the branches and conclusions are rearranged with closer spacing in order to fill in some of the blank space that would be available if the prior spatial arrangement of Fig. 30 were maintained.

To maximize the display of tree objects on a fixed size display, my system first determines how many tree objects to display in one horizontal row of the display. The operator can control the number of tree objects displayed in a horizontal row by using the "Expand" command on the "View" menu to decrease the number of tree objects or the "Reduce" command on the "View" menu to increase the number of tree objects.

Once the number of tree objects in a horizontal row is determined, my system next determines the number of tree objects that can be displayed in a vertical column while maintaining the proper aspect ratio of tree objects. My system

then displays one horizontal row of tree objects at a time without displaying any objects that are beyond the rightmost edge of the display. Any horizontal rows which contain only tree objects beyond the rightmost edge of the display are not displayed. The result of eliminating these rows is that the display surface is more densely packed with at least one tree object in each horizontal row. This eliminates much of the "white space" that would occur when displaying portions of a large tree near the root of the tree.

Fig. 32 illustrates the use of a tree that has as one of its possible conclusions the value of the field for which the tree is being determined. The ability of a tree for a particular field to reference itself is useful in providing the user of the system with values determined by the tree if the tree has anticipated the values of interest. But in the case where the values have not been anticipated by the tree, the self-reference allows the field to be prompted so that the operator can enter the information directly.

Links Tool

In my invention, the Links Tool provides an ability to relate the fields on the form system with the fields in related database(s). Fig. 33 shows the dialog box for establishing both read and write links between applications and the databases. The Links Tool dialog can be entered from either form completion mode or from the Form Tool by using the "Links" command on the "Tools" menu.

The Links Tool dialog shown in Fig. 33 allows the operator to associate database fields (listed on the left side of the dialog box) with fields defined within my form system. This association can be made for both the purpose of reading data from the database and writing data into the database. Fig. 33 is from the Life Insurance Application example used earlier and shows how an applicant's address, city, state, etc. can be obtained from a database given the applicant's name.

Fig. 34 shows the ability of my invention to take care of a case where there is not an established database in place corresponding to the values of the fields within my forms system. In the illustration of Fig. 34, a link named "New Link" has been attempted with a database; in this case, a database table named "New File". The system was unable to open that file because that file did not exist and the option provided in the dialog box allows the operator to create a new database table with this name. My system uses the properties of the fields as defined by the operator to create database fields of the appropriate size and type.

My invention has been described with particular attention to its preferred embodiment; however, it should be understood that variations and modifications within the spirit and scope of the invention may occur to those skilled in the art to which the invention pertains.

APPENDIX A

The following is the file format in which my graphical image data file for documents are saved on disk.

The file is a binary file and can be considered to be a sequence of variable length chunks of data called records. Each record begins with a 2-byte ID data byte followed by 4 bytes define the length of the remainder of the record. The last record of a file is an EOF record.

Multiple-byte data is in little-endian form, i.e., the least significant byte comes first. This is the natural byte order for little-endian machines like those based on the Intel 8088 architecture and its descendants. Implementation of the form system on big-endian machines, like those based on the Motorola 68000 and its offspring, require a byte swap on all multiple-byte data.

Character data and numeric data are in ASCII format.

The only record that contains environment specific information is the FORMPICTURE record. Because an implementation can ignore records with a u2PictureFormat that it does not recognize, picture definitions for multiple environments can coexist, i.e., a file can contain both a Macintosh and a MS Windows version of a picture and as a result be run on either system.

Data Element Naming

In the specification that follows, the name of each data element implies its format on disk. For example, the name u2DummyData, based on its prefix (u2), is a 2 byte unsigned integer with the least significant byte first. Other prefixes are defined in Table 1: Name Prefix Definitions. If a name has no prefix (has in initial capital) it is a complex structure or sequence defined elsewhere.

Table 1: Name Prefix Definitions

Prefix	Meaning
u1	1 byte unsigned integer
u2	2 byte little endian unsigned integer
u4	4 byte little endian unsigned integer
sv	variable length string (u2 length of string followed by string w/o null termination)
dv	variable length data (see separate definition)
ov	variable length object code (see separate definition)

General Data File Format

Every record is organized as shown in Table 2: Record Organization. In the description of individual records the 6 header bytes will not be shown.

Table 2: Record Organization

Name	Comments
u2RecordType	
u4RecordLength	length of data portion
<data portion of record>	

Order of Records

Records of a graphical image data file will always be in the following order; although, some of the records may not be present.

BOF
IGNORE_REMOTE
FORMNAMES
FIELDNAMES
FONTNAMES
for each form

FORMSIZE
for each field, text, picture, or pattern
FORMFIELD, FORMTEXT, FORMPICTURE, or
FORMPATTERN
for each field
FIELDTREE
FIELDHELP
FIELDEXPECT
FIELDVALUE
for each dBase link
DBASE_LINK
for each DDE link
DDE_LINK
for each ASCII link
ASCII_LINK
EOF

Record Definitions

BOF record - beginning of file (type = 1)

Name	Comments
u2ApplicationId	0xA419
u2Version	currently

Description

The BOF must be the first record in every graphical image data file. Borland International may change this number in the future, as the D'BiFF is expanded for future needs.

IGNORE_REMOTE record - ignore remote requests (type = 2)

Name	Comments
u1IgnoreRemoteRequests	1 = ignore remote requests 0 = don't

Description

Flag that causes the application to ignore remote (DDE) requests for data.

FORMNAMES record - form names (type = 3)

Name	Comments
u2NumberOfForms	number of names that follow
svFormName	
. . .	

Description

A form's position in this list of names is its ID, beginning at 1, for use elsewhere in this file.

FIELDNAMES record - field names (type = 4)

Name	Comments
u2NumberOfFields	number of names that follow
svFieldName	
. . .	

Description

A field's position in this list of names is its ID, beginning at 1, for use elsewhere in this file.

FONTNAMES record - font names (type = 5)

Name	Comments
u2NumberOfFonts	number of fonts that
svFontName	follow the number of u2FontSizethe

font size in points.

u1AttributeMask	see Table 3
. . .	

Table 3: Meaning of u1AttributeMask

Bits	Mask	Meaning
7-3	0xF8	Reserved (must be zero)
2	0x04	FONT_UNDERLINE
1	0x02	FONT-ITALIC
0	0x01	FONT_BOLD

Description

A font's position in this list is its ID, beginning at 1, for use elsewhere in this file.

FORMSIZE record - form size (type = 6)

Name	Comments
u2FormId	established in FORMNAMES
u2xSize	units: 1/4 of character width
u2ySize	units: 1/8 of character height

Description

Size of a form.

FORMFIELD record - field on a form (type = 7)

Name	Comments
u2FieldId	established in FIELDNAMES
u2xLoc	units: 1/4 of character width
u2yLoc	units: 1/8 of character height
u2xSize	units: 1/4 of character width
u2ySize	units: 1/8 of character height
PropertyList	last property is always
. . .	EOP

Description

Definition of a field item on the form identified in the last FORMSIZE record.

FORMTEXT record - text on a form (type = 8)

Name	Comments
svText	ASCII text
u2xLoc	units: 1/4 of character width
u2yLoc	units: 1/8 of character height
u2xSize	units: 1/4 of character width
u2ySize	units: 1/8 of character height
PropertyList	last property is always EOP
. . .	

Description

Definition of a text item on the form identified in the last FORMSIZE record.

FORMPICTURE record - picture on a form (type = 9)

Name		Comments
u2xLoc		units: 1/4 of character width
u2yLoc		units: 1/8 of character height
u2xSize		units: 1/4 of character width
u2ySize		units: 1/8 of character height
PictureDefinition		one or more of the following
	u2PictureFormat	0x01 - MS Windows BitMap file
	u4Length	number of bytes that follow
	svFileName	file containing picture
	u2PictureFormat	0x02 = MS Windows Metafile
	u4Length	number of bytes that follow
	svFileName	file containing picture
	u2MapMode	
	u2PictureFormat	otherwise = ignore this record
	u4Length	number of bytes that follow
	<bytes to skip>	
	u2PictureFormat	0x00 = end of picture formats
PropertyList		last property is always EOP
	. . .	

Description

Definition of a picture item on the form identified in the last FORMSIZE record. Each implementation should pick the first picture format it recognizes.

FORMPATTERN record - pattern on a form (type = 10)

Name	Comments
u2xLoc	units: 1/4 of character width
u2yLoc	units: 1/8 of character height
u2xSize	units: 1/4 of character width
u2ySize	units: 1/8 of character height
u1Pattern	0 = horizontal lines
	1 = vertical lines
	2 = diagonal lines, top-left to lower-right

	3 = diagonal lines, lower-left to top-right
	4 = horizontal and vertical lines (cross)
	5 = diagonal lines in both directions (diagonal cross)
	6 = 0% black (white)
	7 = 6% black
	8 = 13% black
	9 = 19% black
	10 = 25% black
	11 = 50% black
	12 = 75% black
	13 = 100% black
	last property is always EOP

PropertyList

. . .

Description

Definition of a pattern item on the form identified in the last FORMSIZE record.

FIELDTREE record - decision tree for a field (type = 11)

Name	Comments
u2FieldId	established in FIELDNAMES
Tree	one or more of the following
(End of tree being last)	
u1NodeDef	Branch node (see Table 4)
ovCondition	
u2FieldId	established in FIELDNAMES
u1NodeDef	Conclusion node (see Table 4)
ovCondition	
ovConclusion	
u1NodeDef	Null node (see Table 4)
ovCondition	
u1NodeDef	End of tree (see Table 4)

Table 4: Meaning of u1NodeDef

Bits	Mask	Meaning
7	0x80	flag: node has a sibling
6	0x40	flag: node has children
5-4	0x30	Reserved (must be zero)
3-0	0x0F	Node type: 0 = End of tree 1 = Branch 2 = Conclusion 3 = Null

Description

The decision tree for a field. The best way to describe the order of the nodes in the file is to show metacode for writing them. To save a tree to disk just pass the top node of the tree to SaveNode().

```
function SaveNode( Node )
    if ( Node )
    {
        SaveNode( Node.FirstChild )
        SaveNode( Node.NextSibling )
        WriteNodeToFile( Node )
    }
```

FIELDHELP record - field specific help (type = 12)

Name	Comments
u2FieldId	established in FIELDNAMES
svHelpText	ASCII help text

Description

Help text for a field

FIELDEXPECT record - field expect (type = 18)

Name	Comments
u2FieldId	established in FIELDNAMES
u2NumberOfValues	number of values that follow dvValue

. . .

Description

This is the list of expected values to be used in a list-box or check-box prompt for the field. The order of the values is maintained.

FIELDVALUE record - field value (type = 13)

Name	Comments
u2FieldId	established in FIELDNAMES
u1ValueSource	0 = User (user input or override) 1 = Circular (user input for circular tree) 2 = Link (external link) 3 = Tree (decision tree)
dvValue	

Description

Value for a field.

DBASE_LINK record - dbase link (type = 19)

Name		Comments
svLinkName		Name for link
svDbaseName		File name for dBase file
u1Inexact		0 = Exact 1 = Inexact
u2NumberOfTriplets		number of triplets that follow
	svDbaseFieldName	Field name
	u2ReadFieldId	established in FIELDNAMES
	u2WriteFieldId	established in FIELDNAME
	. .	
svIndex		File name of index file

Description

This record defines one dBase link.

PDOX_LINK record - Paradox link (type = 20)

Name	Comments
svLinkName	Name of link
svTabName	File name of Paradox table
u1Closest	0 = Not closest
	1 = Closest
u2NumberOfTriplets	number of triplets that follow
svTableFieldName	Field name
u2ReadFieldId	established in FIELDNAMES
u2WriteFieldId	established in FIELDNAMES
. . .	
svIndex	Name for secondary index field

Description

This record defines one Paradox link.

DDE_LINK record - dde link (type = 15)

Name	Comments
svServerApp	Application name
svLinkTopic	Application name
u2NumberOfImports	number of pairs that follow
svRemoteItem	Name in remote application
u2FieldId	established in FIELDNAMES
. . .	

Description

This record defines one DDE link.

ASCII_LINK record - ascii link (type = 16)

Name	Comments
svFileName	File name of ASCII file
u1AccessType	0 = Read
	1 = Write
	2 = Append
u2NumberOfFieldNames	number of names that follow
u2FieldId	established in FIELDNAMES
. . .	

Description

This record defines one ASCII link.

EOF record - end of file (type = 17)

Description

The EOF record must be the last record in the file. It has no data associated with it.

Property Definitions

A series of property definitions is a little like a series of records in which the last property definition is the EOP property, b) the length of a property is implied by its type instead of being specifically declared, and c) the property type is 1 byte long instead of 2.

NOTITLE property - don't display title (type = 1) This property has no data associated with it.

NOOVERRIDE property - don't allow an overridden (type = 2) This property has no data associated with it.

NOTREESHOW property - don't show tree to user (type = 3) This property has no data associated with it.

BORDERMASK property (type = 4)

Names	Comments
u1BorderMask	what borders to display

Bits	Mask	Meaning
7-4	OxFO	Reserved (must be zero)
3	Ox08	BORDER_BOTTOM
2	Ox04	BORDER_TOP
1	Ox02	BORDER_RIGHT
0	Ox01	BORDER_LEFT

The default is to display all borders.

ALIGNMENT (type = 5)

Name	Comments
u1Alignment	tells how to align text 0 = for left alignment 1 = for right alignment 2 = for centered text 3 = for justified text

The default is left alignment.

EOP - end of properties (type = 6)

This property has no data associated with it.

FORMAT_GENERAL (type = 7)

This property has no data associated with it. This property is the default format.

FORMAT_FIXED (type = 8)

Name	Comments
u1Places	decimal places to display

FORMAT_BUSINESS (type = 10)

Name	Comments
u1Places	decimal places to display

FORMAT_CURRENCY (type = 11)

Name	Comments
u1Places	decimal places to display

FORMAT_DATE (type = 12)

Name	Comments
u1DateFormat	0 = mm/dd/yy 1 = mmmm d, yyyy 2 = d-mmm-yy

3 = d-mmm
4 = mmm-yy
5 = hh:mm AM/PM
6 = hh:mm:ss AM/PM
7 = hh:mm
8 = hh:mm:ss
9 = mm/dd/yy hh:mm

FORMAT_LISTBOX (type = 13)
This property has no data associated with it.

FORMAT_CHECKBOX (type = 14)
This property has no data associated with it.

FORMAT_CHECKIF (type = 15)
This property has no data associated with it.

FORMAT_BUTTON (type = 16)
This property has no data associated with it.

FONT (type = 17)

Name	Comments
u2FontId	established in FONTNAMES

FORMAT_SCROLLING (type = 18)
This property has no data associated with it.

FORMAT_PICTURE (type = 19)

Name	Comments
svPictureString	Picture definition string

Variable Length Data

Data is a type byte followed by a variable-length value. Logical and error values are 1 byte long. Text and numeric values are in "sv" format.

More specifically, a data object is one of the following:

Name	Comments
u1DataType svNumber	Ox1A = number the number in ASCII

Name	Comments
u1DataType svText	Ox1B = text the string

Name	Comments
u1DataType u1LogicalValue	Ox1C = logical 0 = No (false) 1 = Yes (true)

Name	Comments

u1DataType
u1ErrorValue
	Ox1d = error
	1 = #DIV/0! (obsolete)
	2 = #Ref! (obsolete)
	2 = #Value! (obsolete)
	4 = NA
	5 = #NAME? (obsolete)
	6 = #NUM! (obsolete)
	7 = #NULL! (obsolete)
	8 = ERR

Object Code (Conclusions and Conditions)

Object code is a sequence of tokens in Reverse Polish order. Some tokens, such as OP_PLUS, are one-bytes tokens; some, such as OPERAND_NAME, have fixed-length information that follows; others, such as OPERAND_TEXT, are followed by variable length data. The data tokens are the same as data objects defined in the section Variable Length Data. Function ID's are listed in Table 5: Function ID's. Here are the details:

Name	Comments
u2CodeLength Code following in Reverse Polish u1TokenType	number of bytes that follow one or more of the OX01 = OP_NEGATION Ox02 = OP_PERCENT OX03 = OP_EXPONENTIATION Ox04 = OP_MULTIPLY Ox05 = OP_DIVIDE Ox06 = OP_PLUS Ox07 = OP_MINUS Ox08 = OP_AMPERSAND Ox09 = OP_EQUAL OxOA = OP_LESS OxOB = OP_GREATER OxOC = OP_LESSEQUAL OxOD = OP_GREATEREQUAL OxOE = OP_NOTEQUAL OxOF = OP_POSITIVE Ox14 = CONTROL_EQUAL

(continued)

Name	Comments
	Ox15 = CONTROL_PARENS
	Ox16 = CONTROL_END_OF_LINE_
u1TokenType	Ox17 = CONTROL_FUNCTION
u2FonctionId	from Table 5
u1ArgumentCount	number of arguments
u1TokenType	Ox18 = OPERAND_NAME
u2FieldId	established in
FIELDNAMES	
u1TokenType	Ox19 = OPERAND_REFERENCE
u2FieldId	established in FIELDNAMES
dvData	Ox1A = OPERAND_NUMBER
	Ox1B = OPERAND_TEXT
	Ox1C = OPERAND_LOGICAL
	Ox1D = OPERAND_ERROR (see Variable Length Data)

Table 5: Function ID's

ID	Function
0x01	@INT
0x02	@DATE
0x03	@DATEVALUE
0x04	@DAY
0x05	@HOUR
0x06	@MINUTE
0x07	@MONTH
0x08	@NOW
0x09	@SECOND
0x0A	@TIME
0x0B	@TIMEVALUE
0x0C	@WEEKDAY
0x0D	@YEAR
0x0E	@ROUND
0x0F	@TYPE
0x10	@SUM
0x11	@MAX
0x12	@MIN
0x22	@CHAR
0x23	@CODE
0x24	@EXACT
0x25	@FIND
0x26	@LEFT
0x27	@LENGTH
0x28	@MID
0x29	@REPLACE
0x30	@REPEAT
0x31	@RIGHT
0x2C	@ABS
0x2D	@MOD
0x2E	@AND
0x2F	@IF
0x30	@NOT
0x31	@OR

0x32	@UPPER
0x33	@LOWER
0x34	@NULL
0x35	@MESSAGE
0x36	@ERR
0x37	@NA
0x38	@PXOPEN
0x39	@CLOSE
0x3A	@TOP
0x3B	@BOTTOM
0x3C	@PREVIOUS
0x3D	@NEXT
0x3E	@CLEAR
0x3F	@DELETE
0x40	UPDATE
0x41	INSERT
0x42	@STORE
0x43	@ASCIIOPEN
0x44	@DDEOPEN

Limits Imposed by this Format

This file definition constrains you to

OBJECT	LIMIT
Forms	65,535
Fields	65,535
Fonts	65,535
Font size	65,535
Nodes in a tree	65,535
X position	16,383 characters
Y position	8,191 characters

Properties Matched to Item Type				
Property	Field	Text	Picture	Pattern
NOTITLE	X	.	.	.
NOOVERRIDE	X	.	.	.
NOTREESHOW	X	.	.	.
BORDERMASK	X	X	X	X
ALIGNMENT	X	X	.	.
FORMAT_GENERAL	X	.	.	.
FORMAT_FIXED	X	.	.	.
FORMAT_PERCENT	X	.	.	.

(continued)

Properties Matched to Item Type				
Property	Field	Text	Picture	Pattern
FORMAT_BUSINESS	X	.	.	.
FORMAT_CURRENCY	X	.	.	.
FORMAT_DATE	X	.	.	.
FORAMT_LISTBOX	X	.	.	.
FORMAT_CHECKBOX	X	.	.	.
FORMAT_CHECKIF	X	.	.	.
FORMAT_BUTTON	X	.	.	.
FORMAT_SCROLLING	X	.	.	.
FORMAT_PICTURE	X	.	.	.
FONT	X	X	.	.
EOP	X	X	X	X
X = Has meaning . = Has no meaning (and is ignored)				

APPENDIX B

APPLICATION PROGRAM OPENING WINDOW (Fig. 5)

File
New - close any open application and prepare for a new application;
Open - open an application from a list of applications currently on the disk;
Resume - resume goal orienting prompting in the goal form after an interruption;
Save - save to the file of the current name;
Save As - Save as a new named file;
Print Form - prints the current form and contents;
Print All - prints all of the forms of a stack;
Exit - return to WINDOWS;
About - display information about form system;

Edit
Undo - undo the last change;
Cut - cut a designated entity and save on clipboard for subsequent use;
Copy - copy a designated entity to a clipboard for subsequent use by the paste command;
Paste - paste an entity from a clipboard to a designated location;
Clear All - clear data from all forms of a stack;

Form
Select - displays a list of forms for selection;
Clear - clears data from the current form only;

Field
Find - prompts for name of field to be found;
Calculate - requests calculation of the field;
Show tree - displays the tree for the field;

View
Screen - presents display in screen format;
Printer- presents display in the printer format;

Tools
Form - select Form tool and select Form Tool Operations from Menu-Items shown in Fig. 6;
Tree - select Tree tool and select Tree Tool Operations from Menu-Items shown in Fig. 7;
Stack - select Stack tool and select Stack Tool Operations from Menu-Items shown in Fig. 8;
Link - follow dialogue windows to create and/or edit links;

FORM TOOL WINDOW OPERATIONS (FIG. 6)

Form — New - Close any open form & prepare for ne form;
Select - Select a form from list of forms;
Find - Find a form with a defined field name;
Close Tool - Close the form tool & return to completion mode;

Edit — Undo - undo the last change;
Cut - cut a designated entity and save on clipboard for subsequent use by paste command;
Copy - copy a designated entity to a clipboard for subsequent use by the paste command;
Paste - paste an entity from a clipboard to a designated location;

Objects — Field - Create a field object, place the field on the form, & set the size of the field;
Text - Create a text object, place the object on the form, & set the size of the object;
Fill Rect - Select a filled rectangle object, place the object on the form, select a hatch pattern, and set the size of the object;
Rounded Rectangle - Select a rounded rectangle object, place the object on the form, select a hatch pattern, and set the size of the object;
Line - Select a line object and place the line on the form;
Graphic - Create a graphic object, place the object on the form, specify the graphic image, and set the size of the object;

Properties — Repeat - Repeat the last selected property;
Field Type

General - text and numerical;
Fixed - numerical with set decimal places;
Percent - numerical only with % display;
Financial - numerical with comma separators;
Currency - numerical with currency symbols;
Date/Time - serial number of date and time since January 1, 1900 - displays date & time;
Scrolling - scroll through field;

True/ False - For field values Yes or No; the field is displayed with YES and NO check boxes;
Button - For fields which default to NO but can be momentarily set to YES;

Picture - define permitted format of entry;

Selection List - For fields with one of several values from a list which is not displayed in the field;
Check Box - For fields with one of several values which are displayed as check boxes in the field; If the field display size is too small to accommodate the boxes, a selection list is displayed when the field is prompted;

Alignment — Left - Left alignment is the default for newly created fields; field values and text objects are displayed at the left edge of the object's display area;
Right - field values and text objects are displayed at the right edge of the object's display area;
Center - field values and text objects are centered in the object's display area;
Justified - Aligns multi-line field values and text objects flush against the object's left and right margins;

Font — Select a font type and font size from a list;

Borders — Outline - This is the default for newly created fields and places lines on all sides of field;
Left - Places vertical line at left edge of object;
Right - Places vertical line at right edge of object;
Top - Places horizontal line at top edge of object; Bottom - Places horizontal line at bottom edge of object;

Fill Pattern — Select a different fill pattern for a selected filled rectangle or a rounded rectangle;

Line Width	Select a different line width for object borders or for lines;
Protection	No override - User cannot enter value in a calculated field; No tree display - Tree is not displayed;
Field	Replace the selected field object with a new field object;
Name/Text	Edit field name;
Help	Attach Help to selected field;
View	Screen - displays screen view; Printer - displays forms as they will appear when printed;
Tools	Tree - Selects Tree tool; Stack - Selects Stack tool; Link - Selects Link tool;

TREE TOOL WINDOW OPERATIONS (FIG. 7)

Tree	Select - Select a tree from a list of trees; Find - Find a tree containing an identified field in a branch, condition, or conclusion; Print - print the current tree; Print all - print all trees; Close tool - close the Tree tool;
Edit	Undo - undo the last change; Cut - cut a designated entity and save on clipboard for subsequent use by paste command; Copy - copy a designated entity to a clipboard for subsequent use by the paste command; Paste - paste an entity from a clipboard to a designated location;
Objects	Branch - Insert a branch object at the same level as the highlighted object (in parallel); Conclusion - Insert a conclusion at the same level as the highlighted object;
Properties	Field - Use a new field or another existing field to replace the field in the current branch object; Condition - Change the condition that selects the current object; Conclusion - For conclusion object-edit expression; Name - For branch object - edit name;
View	Expand - Expand display; Reduce - Reduce display;

STACK TOOL WINDOW OPERATIONS (FIG. 8)

Stack	Close tool - Close the stack tool;
Edit	Undo - undo the last change; Cut - cut a designated entity and save on clipboard for subsequent use by paste comment; Copy - copy a designated entity to a clipboard for subsequent use by the paste command; Paste - paste an entity from a clipboard to a designated location in the stack; Clear All - clear data from all forms of a stack;
Objects	Form - Add a new form to the stack;
Properties	Title - Edit the title of the highlighted form.

Claims

1. A programmable controlled, goal oriented electronic system for form creation and form completion comprising:
means for generating and means for using form data files which define:

a graphical image (13,14,16,18,19,20) of at least one goal oriented form for display on a monitor;
a graphical image (15,24,25,26,30,31,32) of at least one decision tree discretely associated with fields of a form, wherein said at least one decision tree specifies operations to be performed by the system in order to complete a field of said at least one goal oriented form.

2. A system in accordance with claim 1 wherein:

each said decision tree comprises branch objects (24) and conclusion objects (25); and wherein
said objects define logical relations and/or mathematical operations which are the basis for goal oriented prompting within a form and among forms of a set of forms as defined in said form data files.

3. A system in accordance with claim 1 wherein:
said system for generating form data files further comprises:
means for selectively defining data links between selected fields of one or more forms and a variety of different data sources/destinations.

4. A system in accordance with claim 3 wherein:
said data links are selectively defined as being reading and/or writing links.

5. A system in accordance with claim 3 wherein:
said variety of data sources/destinations include: a file of a relational data base; and an ASCII data file.

6. A system in accordance with claim 3 wherein:
said variety of data sources/destinations includes a dynamic data exchange link (DDE) to an application program.

7. A system in accordance with claim 3 wherein:
said system comprises means for detecting a request for a link to a non-existent data source/destination; and means for creating a data base in which the fields correspond in name and characteristics to the fields named in said link request.

8. A system in accordance with claim 1 wherein:
said means for generating comprises a form tool and a tree tool.

9. A system in accordance with claim 3 wherein:
said means for defining data links comprises a link tool.

10. A system in accordance with claim 1 wherein:
said system comprises a form creation mode of operation for generating and using said graphical images; and a run time mode of operation with facilities limited to use, but not alteration, of said form data files.

11. A system in accordance with claim 9 wherein:
said run time mode of operation comprises means for selecting a field of a form; and means for selectively displaying a decision tree assigned to that field.

Patentansprüche

1. Programmgesteuertes, zielorientiertes, elektronisches System zum Erzeugen und Vervollständigen von Formblättern, umfassend:
eine Einrichtung zum Erzeugen und eine Einrichtung zum Verwenden von Formblatt-Datendateien, welche definieren:

ein graphisches Bild (13, 14, 16, 18, 19, 20) von wenigstens einem zielorientierten Formblatt zum Anzeigen an einem Monitor;
ein graphisches Bild (15, 24, 25, 26, 30, 31, 32) wenigstens eines den Feldern eines Formblatts diskret zugeordneten Entscheidungsbaums, wobei der wenigstens eine Entscheidungsbaum die Operationen im einzelnen angibt, welche durch das System ausgeführt werden, um ein Feld des wenigstens einen zielorientierten Formblattes zu vervollständigen.

2. System gemäß Anspruch 1,

bei welchem jeder der Entscheidungsbäume Abzweigobjekte (24) und Folgerungsobjekte (25) umfaßt; und wobei die Objekte logische Beziehungen und/oder mathematische Operationen definieren, welche innerhalb eines Formblatts und zwischen Formblättern eines Satzes von Formblättern, wie sie in den Formblatt-Datendateien festgelegt sind, die Grundlage für eine zielorientierte Bedienerführung sind.

3. System gemäß Anspruch 1,
wobei das System zum Erzeugen von Formblatt-Datendateien ferner umfaßt:
eine Einrichtung zum selektiven Festlegen von Datenverknüpfungen zwischen ausgewählten Feldern eines oder mehrerer Formblätter und einer Vielzahl von verschiedenen Datenquellen/Datenzielen.

4. System gemäß Anspruch 3,
wobei die Datenverknüpfungen selektiv als Leseverknüpfungen und/oder als Schreibverknüpfungen festgelegt sind.

5. System gemäß Anspruch 3,
wobei die Vielzahl von Datenquellen/Datenzielen enthalten:
eine Datei einer relationalen Datenbank; und eine ASCII-Datendatei.

6. System gemäß Anspruch 3,
wobei die Vielzahl von Datenquellen/Datenzielen eine dynamische Datenaustauschverknüpfung (Dynamic Data Exchange Link, DDE) an ein Anwendungsprogramm enthalten.

7. System gemäß Anspruch 3,

wobei das System eine Einrichtung zum Erfassen einer Anforderung für eine Verknüpfung mit einer nicht vorhandenen Datenquelle/einem nicht vorhandenen Datenziel umfaßt; und
eine Einrichtung zum Erzeugen einer Datenbank, in welcher die Felder bezüglich des Namens und den Kenndaten den in der Verknüpfungsanforderung benannten Feldern entsprechen.

8. System gemäß Anspruch 1,
wobei die Einrichtung zum Erzeugen ein Formblattwerkzeug und ein Baumwerkzeug umfaßt.

9. System gemäß Anspruch 3,
wobei die Einrichtung zum Festlegen der Datenverknüpfungen ein Verknüpfungs-Werkzeug umfaßt.

10. System gemäß Anspruch 1,

wobei das System eine Formblatt-Erzeugungsbetriebsart zum Erzeugen und Verwenden der graphischen Bilder umfaßt; und
eine Laufzeit-Betriebsart umfaßt, welche auf das Verwenden beschränkt aber nicht für das Ändern der Formblattdaten-Dateien Unterstützung bietet.

11. System gemäß Anspruch 9,

wobei die Laufzeit-Betriebsart eine Einrichtung zum Auswählen eines Feldes eines Formblatts umfaßt; und
eine Einrichtung zum selektiven Anzeigen eines Entscheidungsbaums, welcher diesem Feld zugewiesen ist.

Revendications

1. Système électronique commandé programmable, dédié à la création et au remplissage de formulaires, comportant:
des moyens pour générer et des moyens pour utiliser des fichiers de données de formulaires qui définissent :

une image graphique (13, 14, 16, 18, 19, 20), d'au moins un formulaire dédié, à afficher sur un moniteur;
une image graphique (15, 24, 25, 26, 30, 31, 32) d'au moins un arbre de décision associé individuellement à des champs d'un formulaire, dans lequel ledit arbre de décision au nombre d'au moins un spécifie des opérations à exécuter par le système afin de remplir un champ dudit formulaire dédié, au nombre d'au moins un.

2. Système selon la revendication 1, dans lequel :

chaque dit arbre de décision comporte des objets de branchement (24) et des objets de conclusion (25) ; et dans lequel
lesdits objets définissent des relations logiques et/ou des opérations mathématiques qui constituent la base d'un guidage dédié à l'intérieur d'un formulaire et parmi des formulaires d'un ensemble de formulaires, tel que défini dans lesdits fichiers de données de formulaires.

3. Système selon la revendication 1, dans lequel :
ledit système pour générer des fichiers de données de formulaires comporte, en outre :
des moyens pour définir sélectivement des liens de données entre des champs sélectionnés d'un ou de plusieurs formulaires et un éventail de différentes sources / destinations de données.

4. Système selon la revendication 3, dans lequel :
lesdits liens de données sont définis sélectivement comme liens de lecture et/ou d'écriture.

5. Système selon la revendication 3, dans lequel :
ledit éventail de sources / destinations de données comprend : un fichier d'une base de données relationnelle ; et un fichier de données ASCII.

6. Système selon la revendication 3, dans lequel :
ledit éventail de sources / destinations de données comprend un lien d'échange dynamique de données (DDE) avec un programme d'applications.

7. Système selon la revendication 3, dans lequel ledit système comprend des moyens pour détecter une demande d'un lien avec une source / destination de données inexistante ; et des moyens pour créer une base de données dans laquelle les champs correspondent en nom et en caractéristiques aux champs nommés dans ladite demande de lien.

8. Système selon la revendication 1, dans lequel :
lesdits moyens de génération comportent un outil pour formulaires et un outil pour arbres.

9. Système selon la revendication 3, dans lequel :
lesdits moyens de définition de liens de données comportent un outil pour liens.

10. Système selon la revendication 1, dans lequel :
ledit système comporte un mode de fonctionnement en création de formulaires pour générer et utiliser lesdites images graphiques ; et un mode de fonctionnement en exécution ayant des possibilités limitées à l'utilisation, mais n'autorisant pas la modification, desdits fichiers de données de formulaires .

11. Système selon la revendication 9, dans lequel :
ledit mode de fonctionnement en exécution comporte des moyens pour sélectionner un champ d'un formulaire; et des moyens pour afficher sélectivement un arbre de décision affecté audit champ.

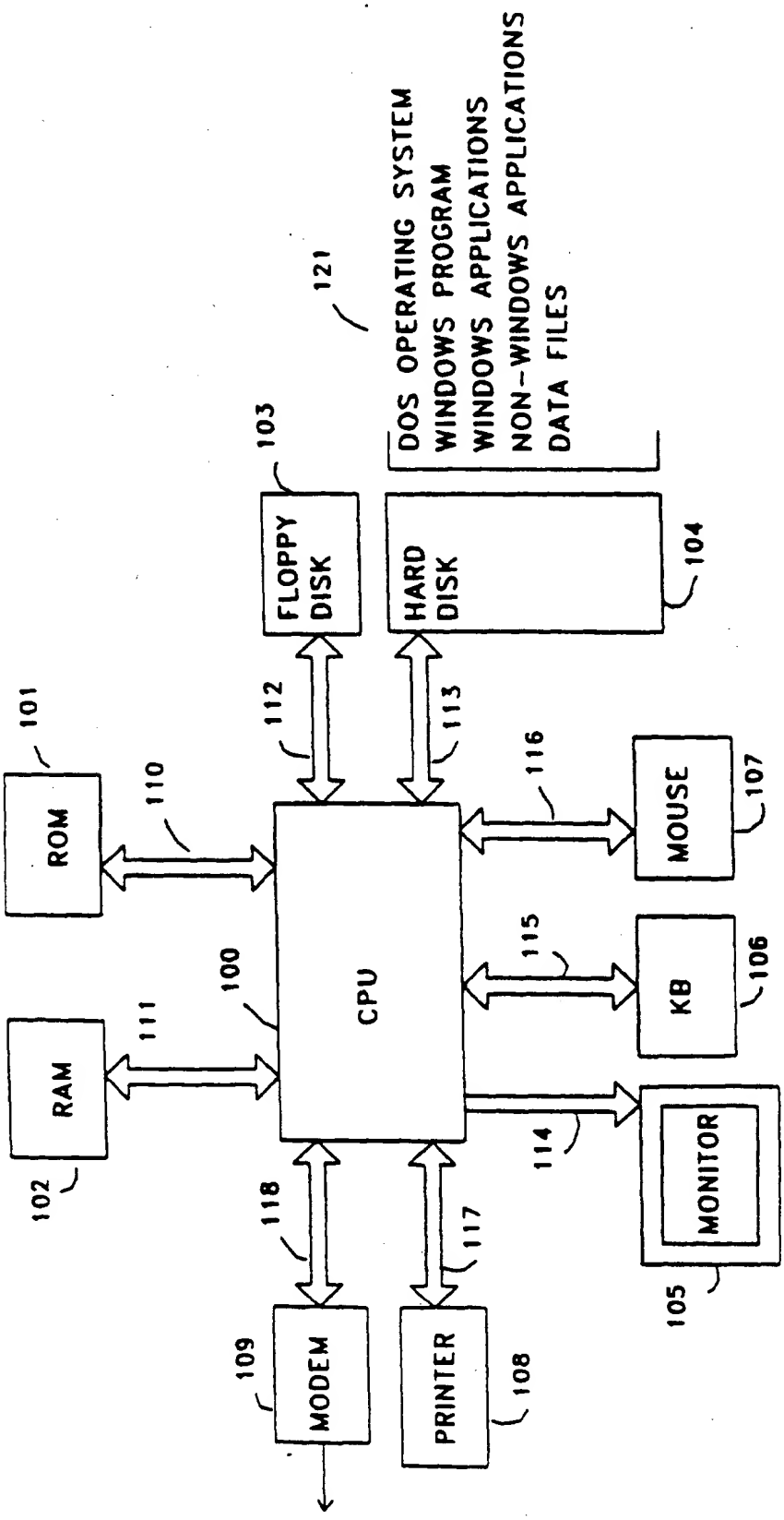
DOS OPERATING SYSTEM
WINDOWS PROGRAM
WINDOWS APPLICATIONS
NON-WINDOWS APPLICATIONS
DATA FILES
121
FLOPPY DISK
103
HARD DISK
104
ROM
101
110
112
113
116
MOUSE
107
CPU
100
115
KB
106
RAM
111
102
114
MONITOR
105
118
117
MODEM
109
PRINTER
108

Fig. 1

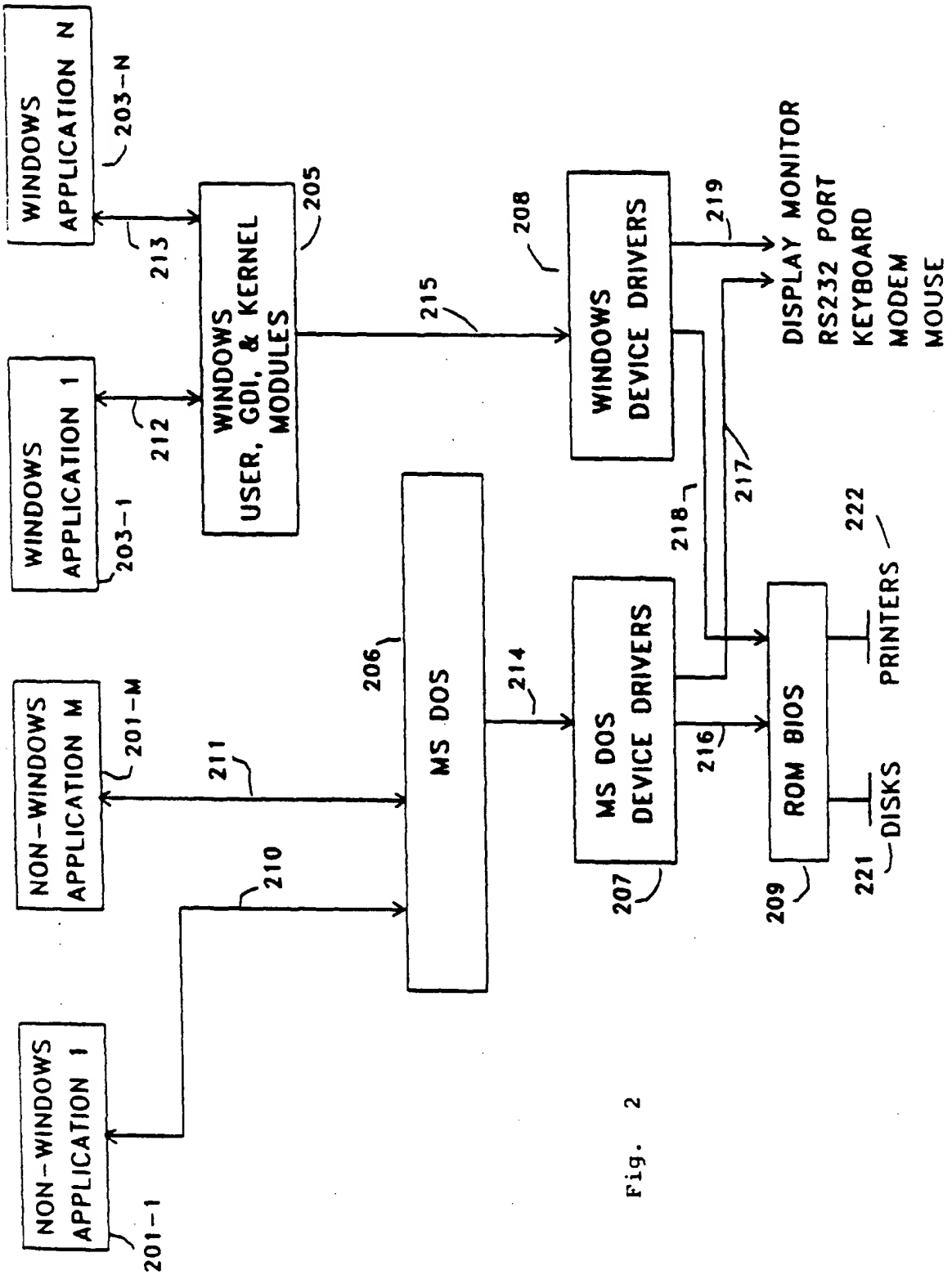
WINDOWS APPLICATION N
203-N
213
WINDOWS APPLICATION 1
203-1
212
WINDOWS USER, GDI, & KERNEL MODULES
205
215
208
WINDOWS DEVICE DRIVERS
219
DISPLAY MONITOR
RS232 PORT
KEYBOARD
MODEM
MOUSE
NON-WINDOWS APPLICATION M
201-M
211
206
MS DOS
214
MS DOS DEVICE DRIVERS
218
217
222
PRINTERS
216
NON-WINDOWS APPLICATION 1
201-1
210
207
209
ROM BIOS
221
DISKS

Fig. 2

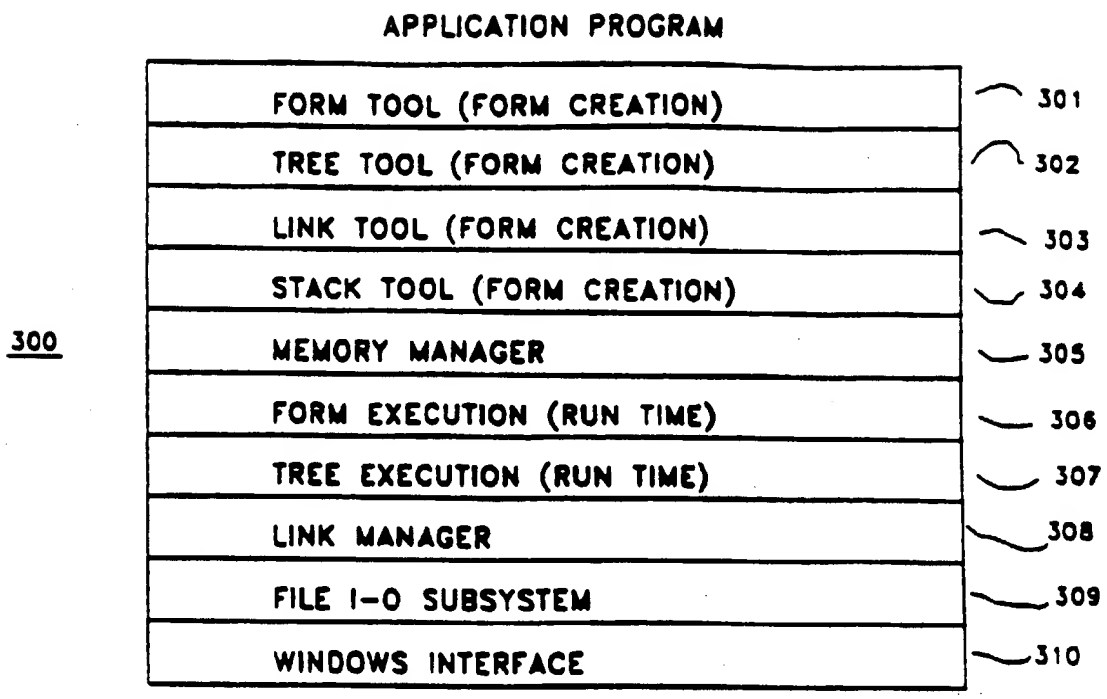
APPLICATION PROGRAM
FORM TOOL (FORM CREATION)
TREE TOOL (FORM CREATION)
LINK TOOL (FORM CREATION)
STACK TOOL (FORM CREATION)
MEMORY MANAGER
FORM EXECUTION (RUN TIME)
TREE EXECUTION (RUN TIME)
LINK MANAGER
FILE I-O SUBSYSTEM
WINDOWS INTERFACE
300
301
302
303
304
305
306
307
308
309
310

Fig. 3

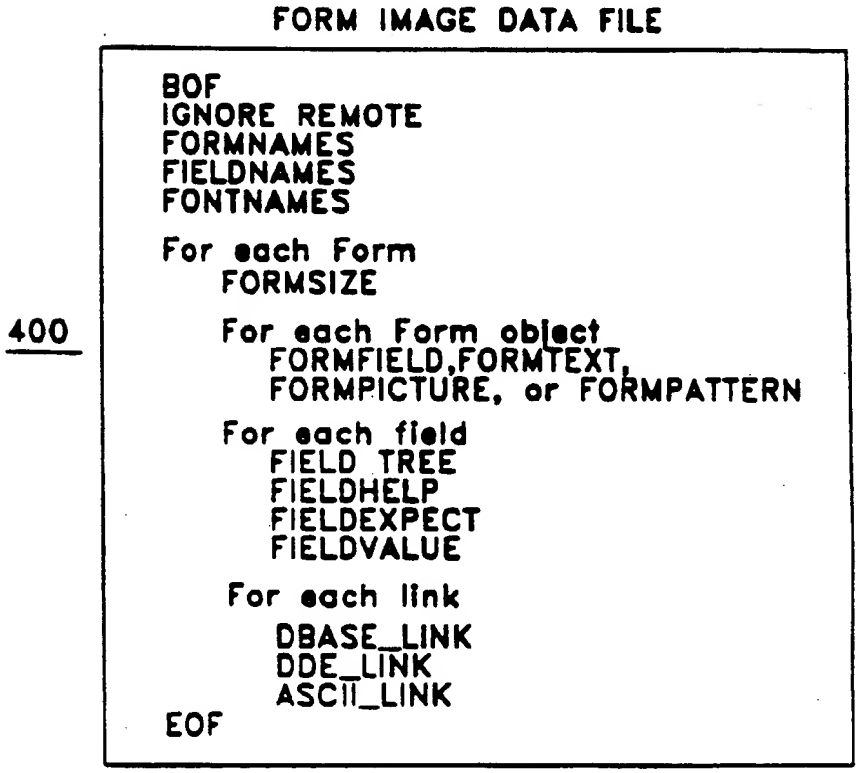
FORM IMAGE DATA FILE
BOF
IGNORE REMOTE
FORMNAMES
FIELDNAMES
FONTNAMES
For each Form
FORMSIZE
For each Form object
FORMFIELD,FORMTEXT,
FORMPICTURE, or FORMPATTERN
For each field
FIELD_TREE
FIELDHELP
FIELDEXPECT
FIELDVALUE
For each link
DBASE_LINK
DDE_LINK
ASCII_LINK
EOF
400

Fig. 4

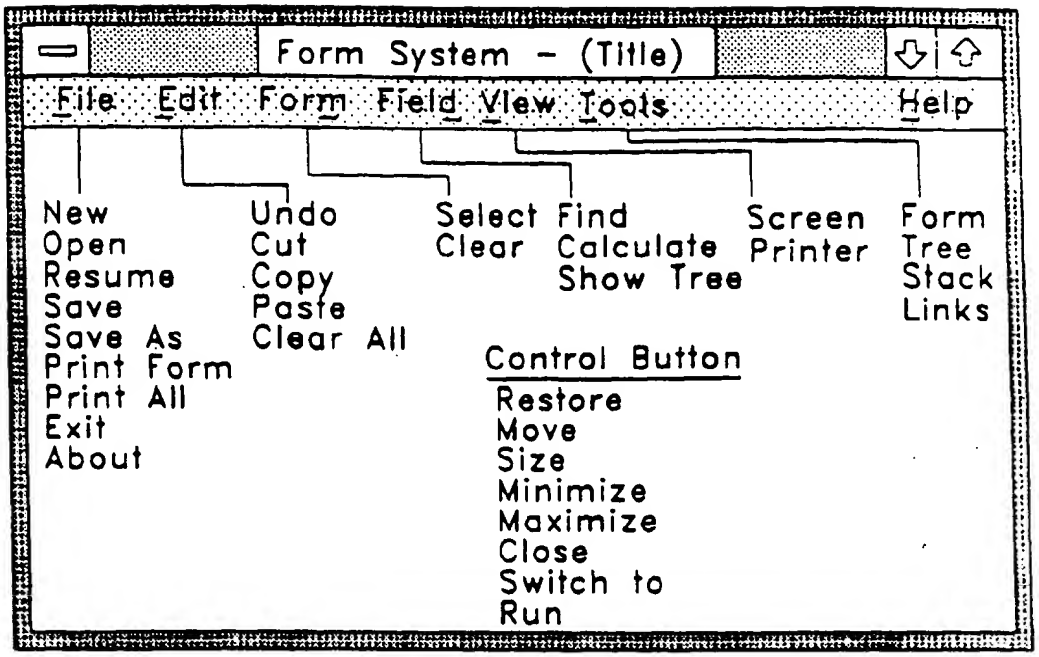
Form System - (Title)
File Edit Form Field View Tools Help
New
Open
Resume
Save
Save As
Print Form
Print All
Exit
About
Undo
Cut
Copy
Paste
Clear All
Select
Clear
Find
Calculate
Show Tree
Screen
Printer
Form
Tree
Stack
Links
Control Button
Restore
Move
Size
Minimize
Maximize
Close
Switch to
Run

Fig. 5

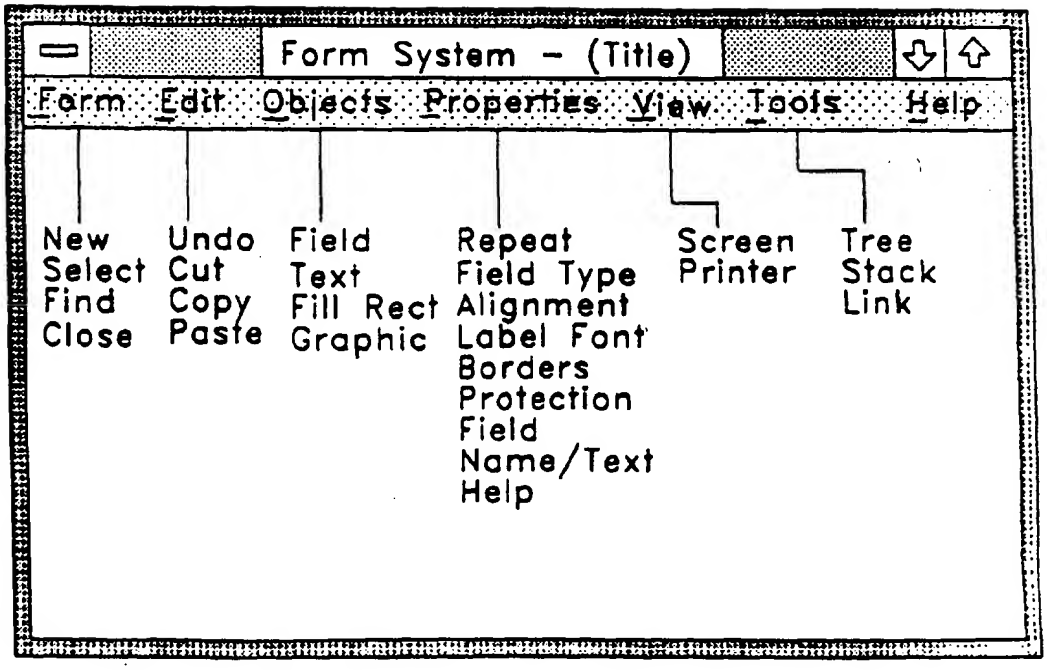
Form System - (Title)
Form Edit Objects Properties View Tools Help
New
Select
Find
Close
Undo
Cut
Copy
Paste
Field
Text
Fill Rect
Graphic
Repeat
Field Type
Alignment
Label Font
Borders
Protection
Field
Name/Text
Help
Screen
Printer
Tree
Stack
Link

Fig. 6

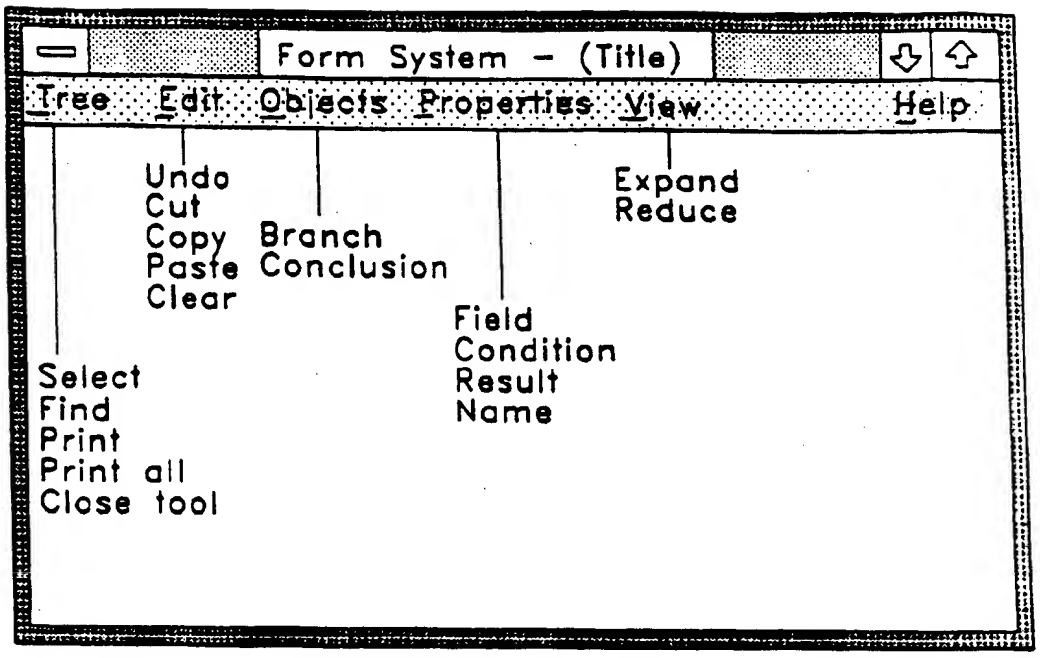
Form System - (Title)
Tree Edit Objects Properties View Help
Undo
Cut
Copy
Paste
Clear
Branch
Conclusion
Field
Condition
Result
Name
Expand
Reduce
Select
Find
Print
Print all
Close tool

Fig. 7

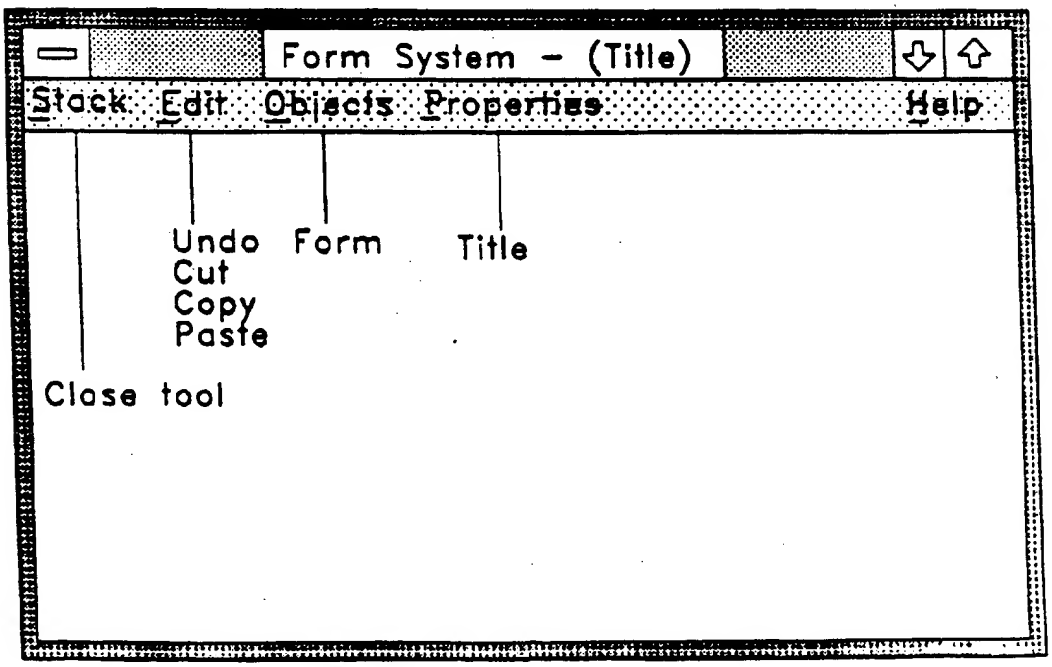
Form System - (Title)
Stack Edit Objects Properties Help
Undo
Cut
Copy
Paste
Form
Title
Close tool

Fig. 8

Apex Life Insurance Company

Proposed Insured				
Residence address		City	State	Zip code
Sex ☐ M ☐ F	Birthdate	Birthplace	Residence telephone	
Beneficiary name			Relationship to Insured	
Beneficiary address				
Total annual premium	Premium payment amount			
☐ Insured does not meet basic qualifications		☐ Temporary insurance not available		
☐ Insured may be subject to substandard rating		☐ Policy may require exclusion rider		
☐ Medical exam required		Deposit required	Deposit received	
Signature >				

Fig. 9

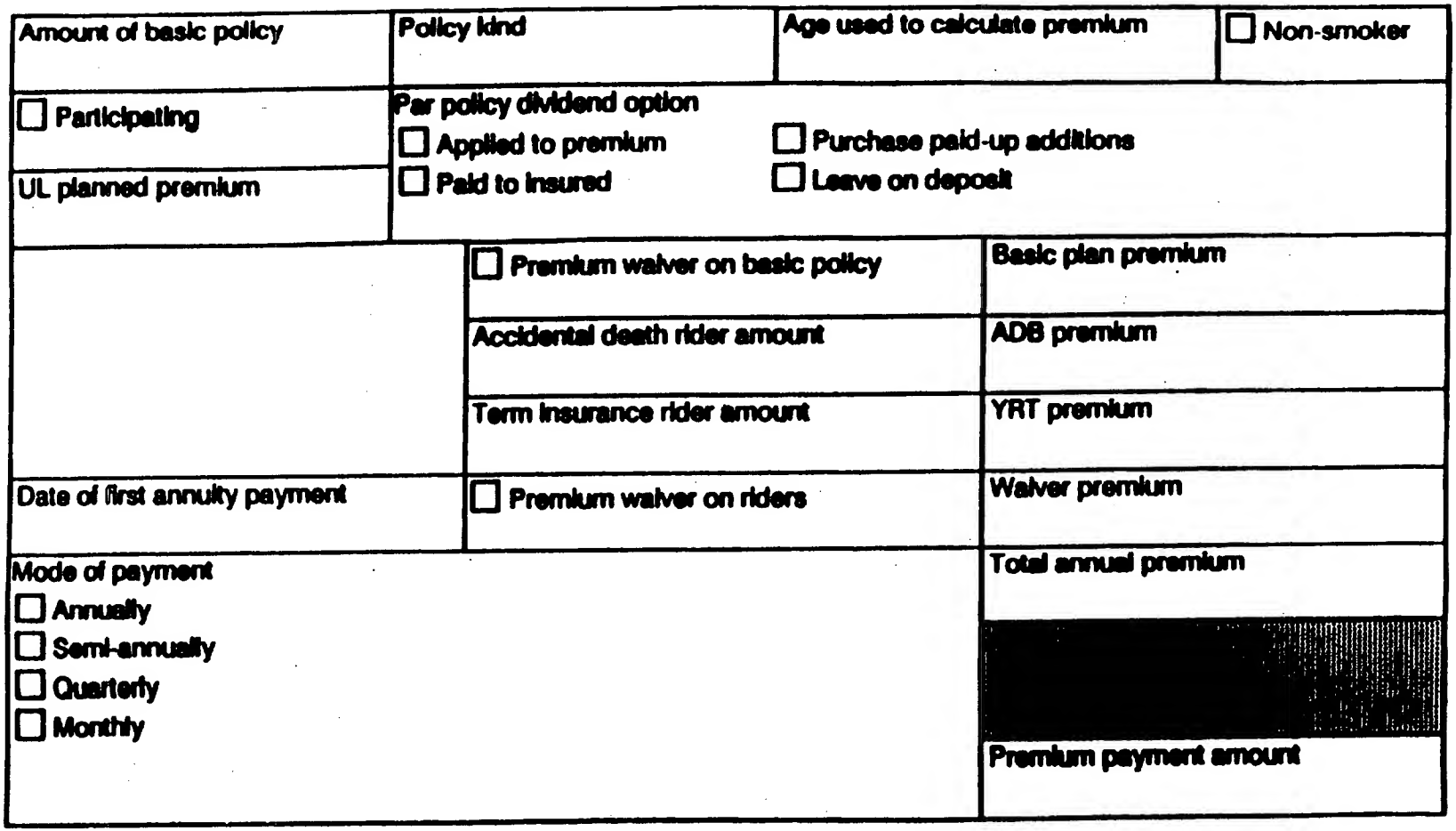

Fig. 10

EP 0 483 664 B1

Have you: In the past 12 months had any known or suspected heart attack, stroke, or cancer, other than of the skin, or been treated by any physician or other practitioner for any of these conditions?	☐ Yes ☐ No
Within the last 60 days been advised by any physician or other practitioner to have any diagnostic test or surgery not yet performed?	☐ Yes ☐ No
Have you smoked cigarettes in the last 12 months?	☐ Yes ☐ No
Have you used tobacco in any other form in the last 12 months?	☐ Yes ☐ No
Will any existing life or annuity coverage be replaced, lapsed or surrendered?	☐ Yes ☐ No
Do you have any other application pending for life insurance?	☐ Yes ☐ No
Are you in the Reserves, National Guard, on active duty in the military, or enrolled in a college military program?	☐ Yes ☐ No
Have you in the last three years engaged in or do you plan to engage in any of the following activities? ☐ Motorized vehicle racing ☐ Mountain climbing ☐ Scuba diving	

Fig. 11

EP 0 483 664 B1

Height (inches)	Weight	Has your weight changed more than 10 pounds in the last year? ☐ Yes ☐ No
Are you at the present time taking any medications?		☐ Yes ☐ No
Are you presently under a doctor's care for any condition?		☐ Yes ☐ No
Have you ever had any operations?		☐ Yes ☐ No
Have any operations ever been advised but not performed?		☐ Yes ☐ No
Do you have any impairment of sight or hearing?		☐ Yes ☐ No
Have you had an electrocardiogram or x-ray made in the last five years?		☐ Yes ☐ No
Has a parent or sibling ever had heart disease, high blood pressure or diabetes?		☐ Yes ☐ No
Remarks		

Fig. 12

EP 0 483 664 B1

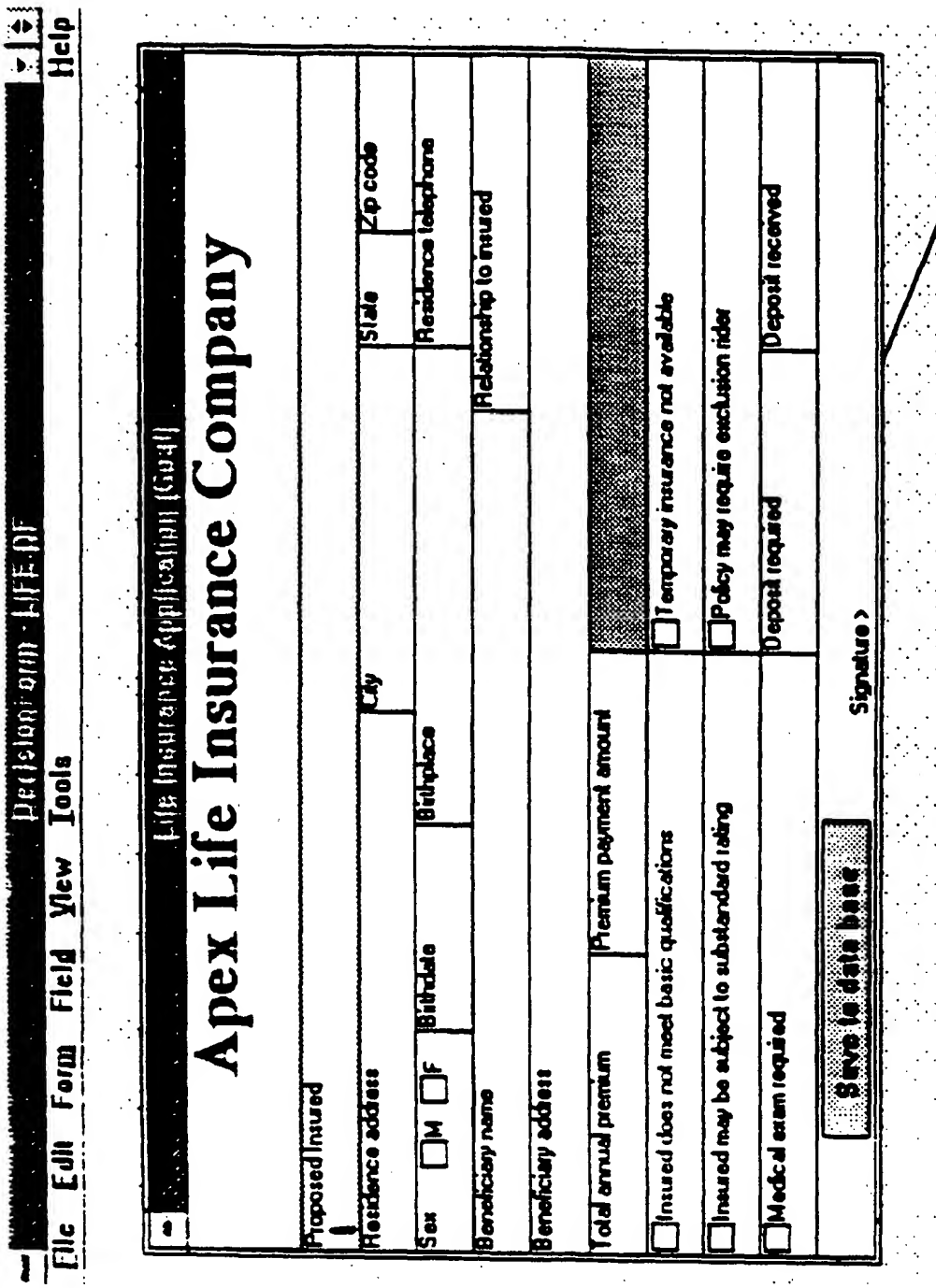
Decision Plus - LIFE.DF
File Edit Form Field View Tools Help
Life Insurance Application (Gen1)
Apex Life Insurance Company
Proposed Insured
Residence address
City
State
Zip code
Sex M F
Birthdate
Birthplace
Residence telephone
Beneficiary name
Relationship to insured
Beneficiary address
Total annual premium
Premium payment amount
Insured does not meet basic qualifications
Temporary insurance not available
Insured may be subject to substandard rating
Policy may require exclusion rider
Medical exam required
Deposit required
Deposit received
Save to data base
Signature
13

Fig. 13

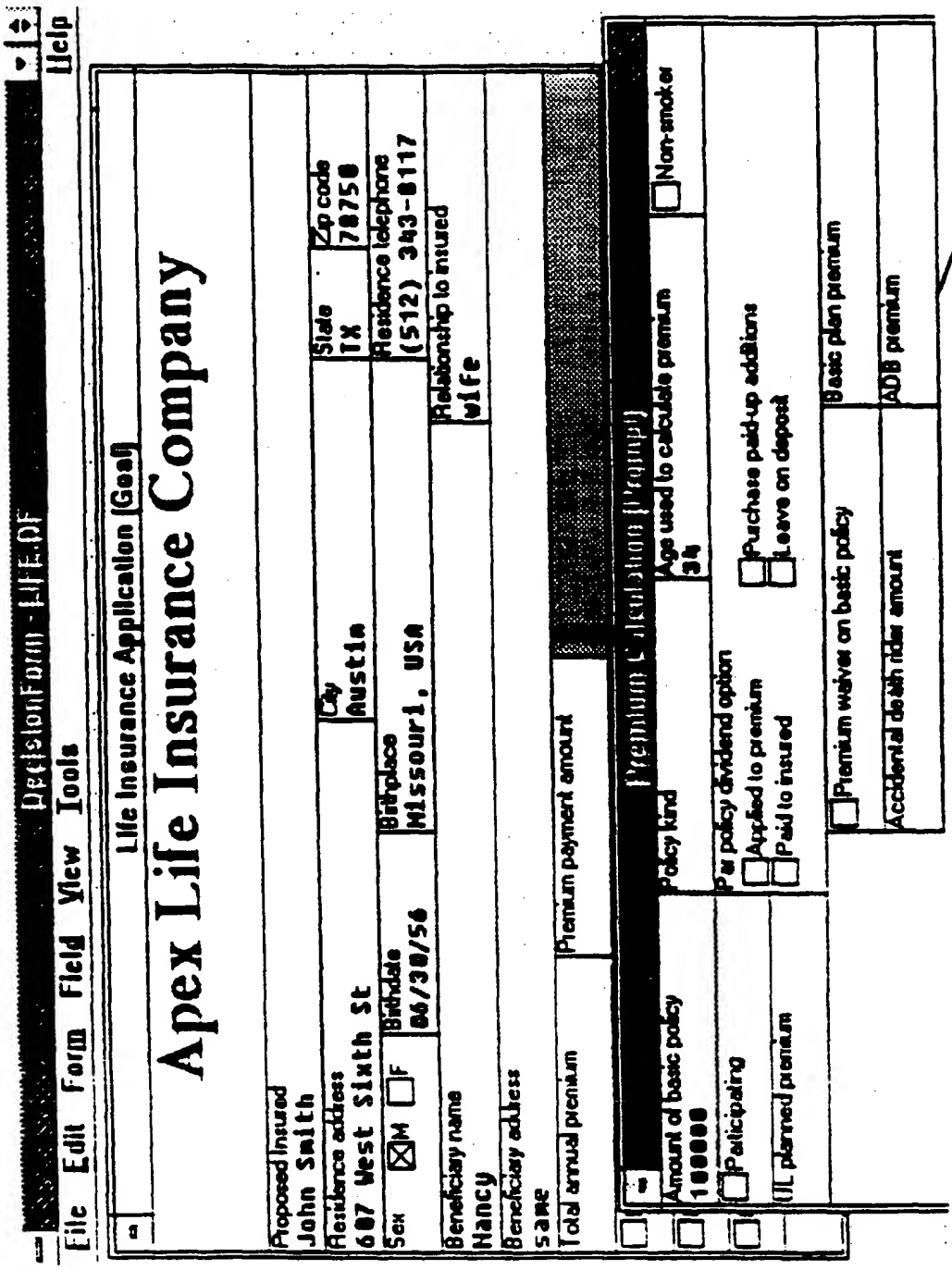
DecisionForm - LIFE.DF
File Edit Form Field View Tools Help
Life Insurance Application (Goal)
Apex Life Insurance Company
Proposed insured
John Smith
Residence address
607 West Sixth St
City
Austin
State
TX
Zip code
78758
Sex M F
Birthdate
06/30/56
Birthplace
Missouri, USA
Residence telephone
(512) 343-8117
Beneficiary name
Nancy
Relationship to insured
wife
Beneficiary address
same
Total annual premium
Premium payment amount
Premium calculation (Prompt)
Amount of basic policy
100000
Policy kind
Age used to calculate premium
34
Non-smoker
Participating
Par policy dividend option
Applied to premium
Paid to insured
Purchase paid-up additions
Leave on deposit
UL planned premium
Premium waives on basic policy
Basic plan premium
Accidental death rider amount
ADB premium
14

Fig. 14

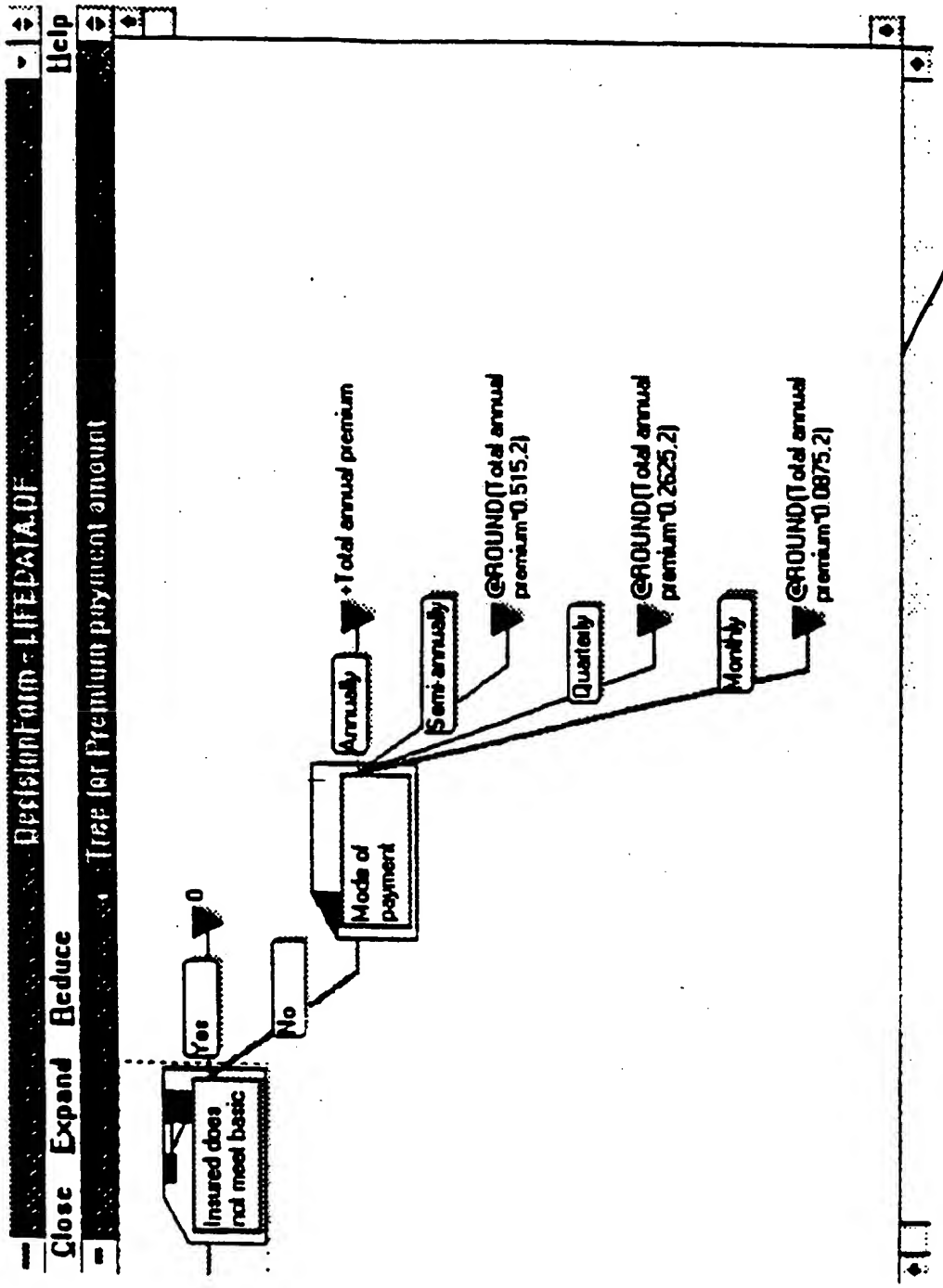
DecisionForm - LIFEDATA.DF
Help
Close Expand Reduce
Tree for Premium payment amount
Insured does not meet basic
Yes
No
0
Mode of payment
Annually
Total annual premium
Semi-annually
@ROUND(Total annual premium*0.515,2)
Quarterly
@ROUND(Total annual premium*0.2625,2)
Monthly
@ROUND(Total annual premium*0.0875,2)
15

Fig. 15

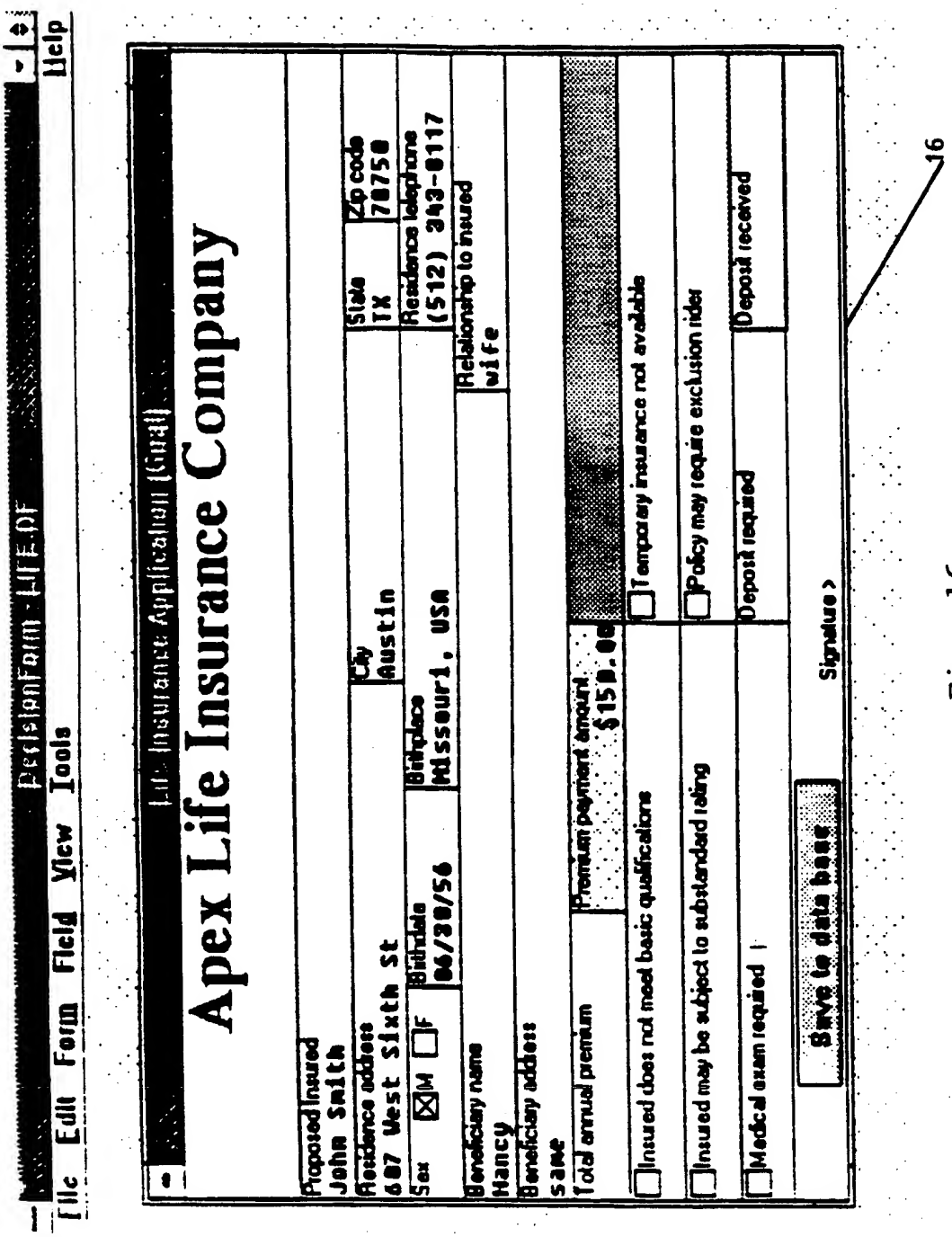
PerfectForm - LIFE.DF
File Edit Form Field View Tools Help
Life Insurance Application (Gmail)
Apex Life Insurance Company
Proposed Insured
John Smith
Residence address
687 West Sixth St
City
Austin
State
TX
Zip code
78758
Sex ☒M ☐F
Birthdate
06/30/56
Birthplace
Missouri, USA
Residence telephone
(512) 343-8117
Beneficiary name
Nancy
Relationship to insured
wife
Beneficiary address
same
Total annual premium
Premium payment amount
$150.00
Insured does not meet basic qualifications
Temporary insurance not available
Insured may be subject to substandard rating
Policy may require exclusion rider
Medical exam required
Deposit required
Deposit received
Save to data base
Signature
16

Fig. 16

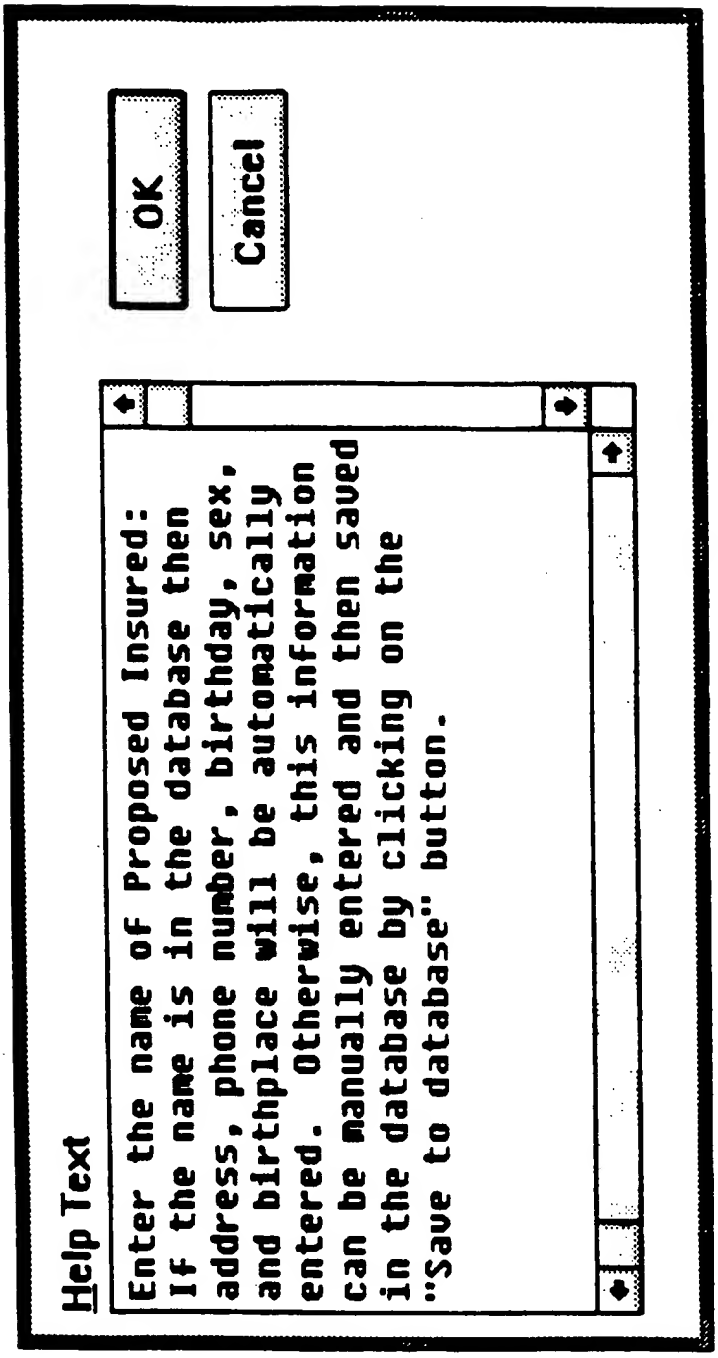
Help Text
Enter the name of Proposed Insured:
If the name is in the database then
address, phone number, birthday, sex,
and birthplace will be automatically
entered. Otherwise, this information
can be manually entered and then saved
in the database by clicking on the
"Save to database" button.
OK
Cancel

Fig. 17

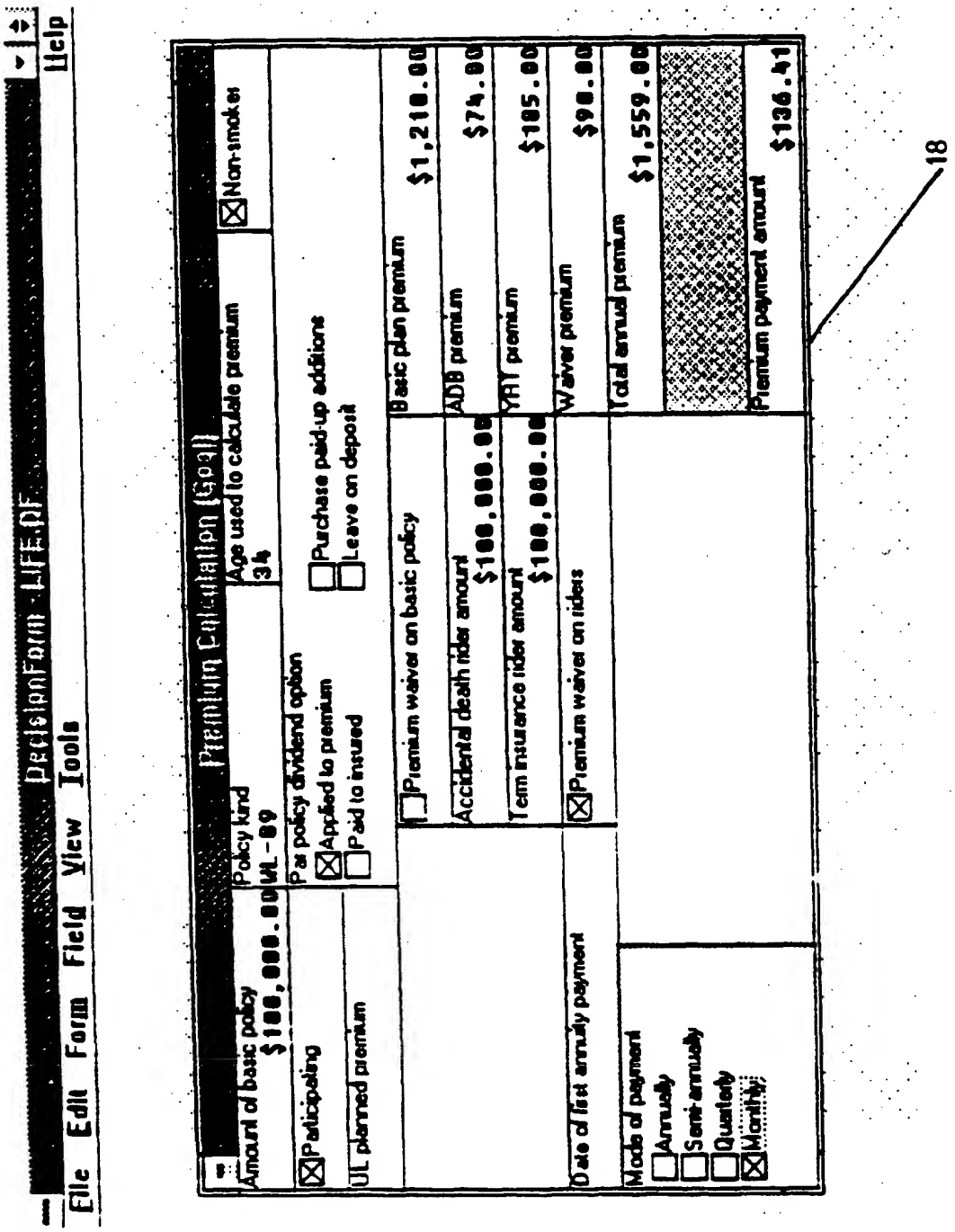
DecisionForm - LIFE.DF
File Edit Form Field View Tools Help
Premium Calculation [Goal]
Amount of basic policy
$100,000.00
Policy kind
WL-09
Age used to calculate premium
34
Non-smoker
Par policy dividend option
Participating
Applied to premium
Paid to insured
Purchase paid-up additions
Leave on deposit
UL planned premium
Premium waiver on basic policy
Basic plan premium
$1,210.00
Accidental death rider amount
$100,000.00
ADB premium
$74.00
Term insurance rider amount
$100,000.00
YRT premium
$185.00
Date of first annuity payment
Premium waiver on rides
Waiver premium
$98.00
Mode of payment
Annually
Semi-annually
Quarterly
Monthly
Total annual premium
$1,559.00
Premium payment amount
$136.41
18

Fig. 18

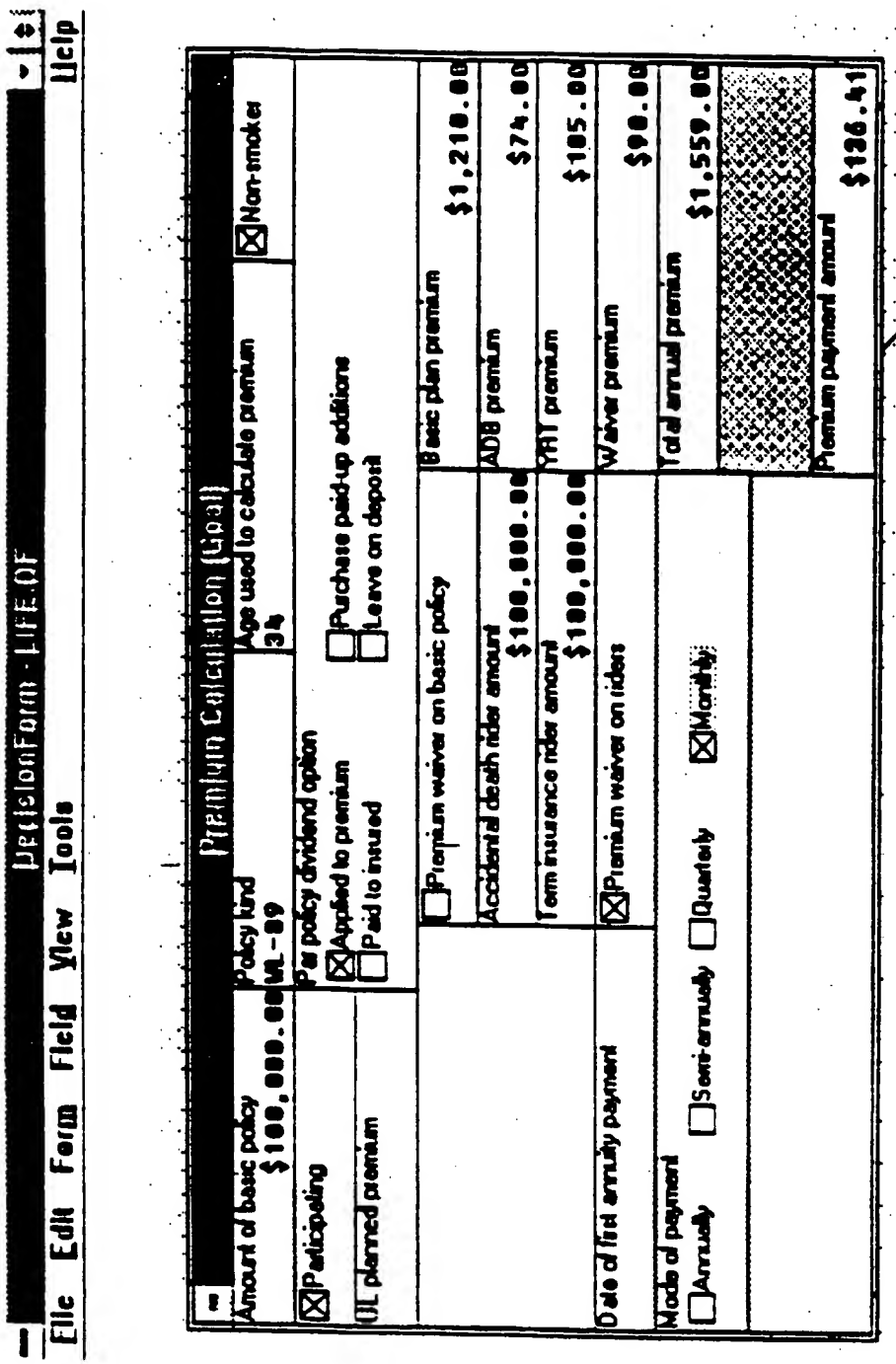
PolicyIllustrationForm - LIFE.DF
File Edit Form Field View Tools Help
Premium Calculation (Goal)
Amount of basic policy
$100,000.00
Policy kind
WL-89
Age used to calculate premium
34
Non-smoker
Participating
UL planned premium
Par policy dividend option
Applied to premium
Paid to insured
Purchase paid-up additions
Leave on deposit
Premium waiver on basic policy
Accidental death rider amount
$100,000.00
Term insurance rider amount
$100,000.00
Basic plan premium
$1,210.00
ADB premium
$74.00
PWT premium
$105.00
Waiver premium
$90.00
Total annual premium
$1,559.00
Premium payment amount
$136.41
Date of first annuity payment
Premium waiver on riders
Mode of payment
Annually
Semi-annually
Quarterly
Monthly
19

Fig. 19

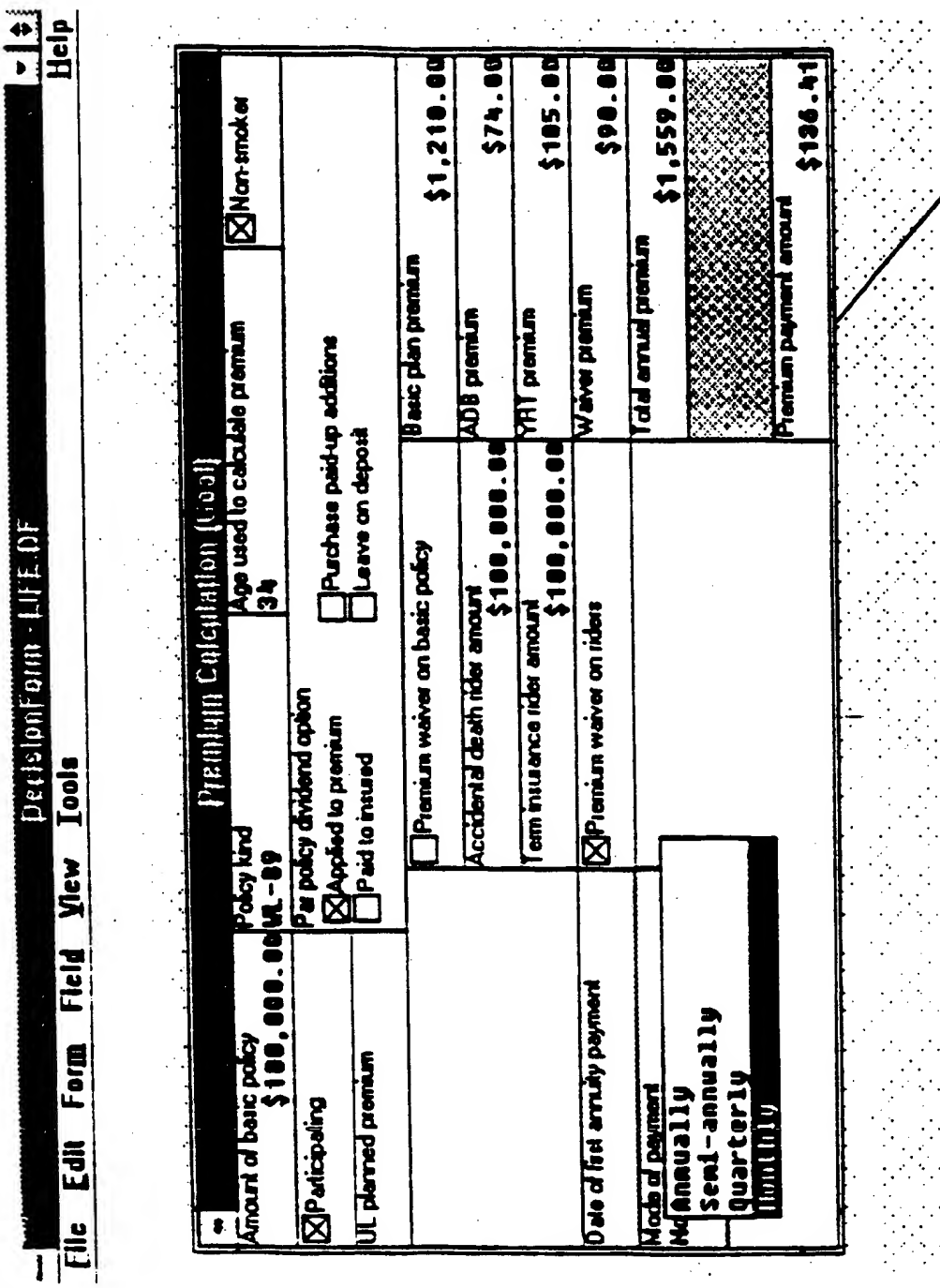
DesignForm - LIFE.DF
File Edit Form Field View Tools Help
Premium Calculation (Tool)
Amount of basic policy
$100,000.00
Policy kind
UL-10
Age used to calculate premium
34
Non-smoker
Par policy dividend option
Participating
Applied to premium
Paid to insured
Purchase paid-up additions
Leave on deposit
UL planned premium
Premium waiver on basic policy
Basic plan premium
$1,210.00
Accidental death rider amount
$100,000.00
ADB premium
$74.00
Term insurance rider amount
$100,000.00
TRT premium
$105.00
Date of first annuity payment
Premium waiver on riders
Waiver premium
$90.00
Mode of payment
Annually
Semi-annually
Quarterly
Monthly
Total annual premium
$1,559.00
Premium payment amount
$136.41
20

Fig. 20

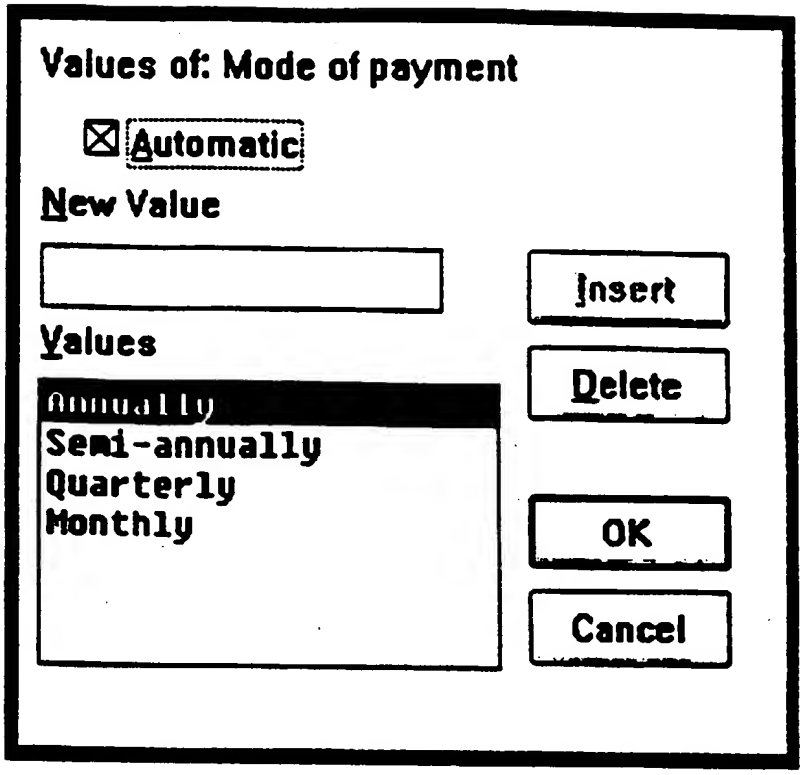
Values of: Mode of payment
Automatic
New Value
Insert
Values
Delete
Annually
Semi-annually
Quarterly
Monthly
OK
Cancel

Fig. 21

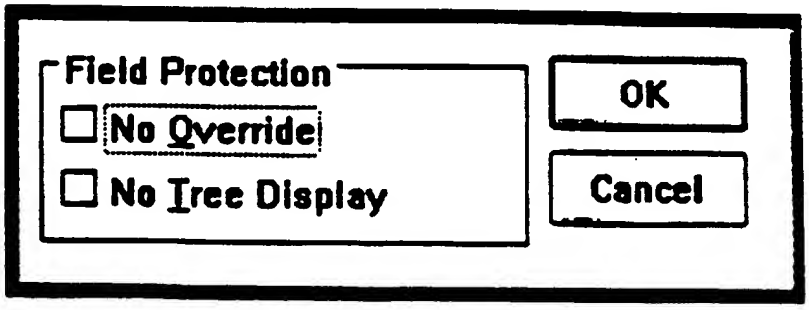
Field Protection
No Override
No Tree Display
OK
Cancel

Fig. 22

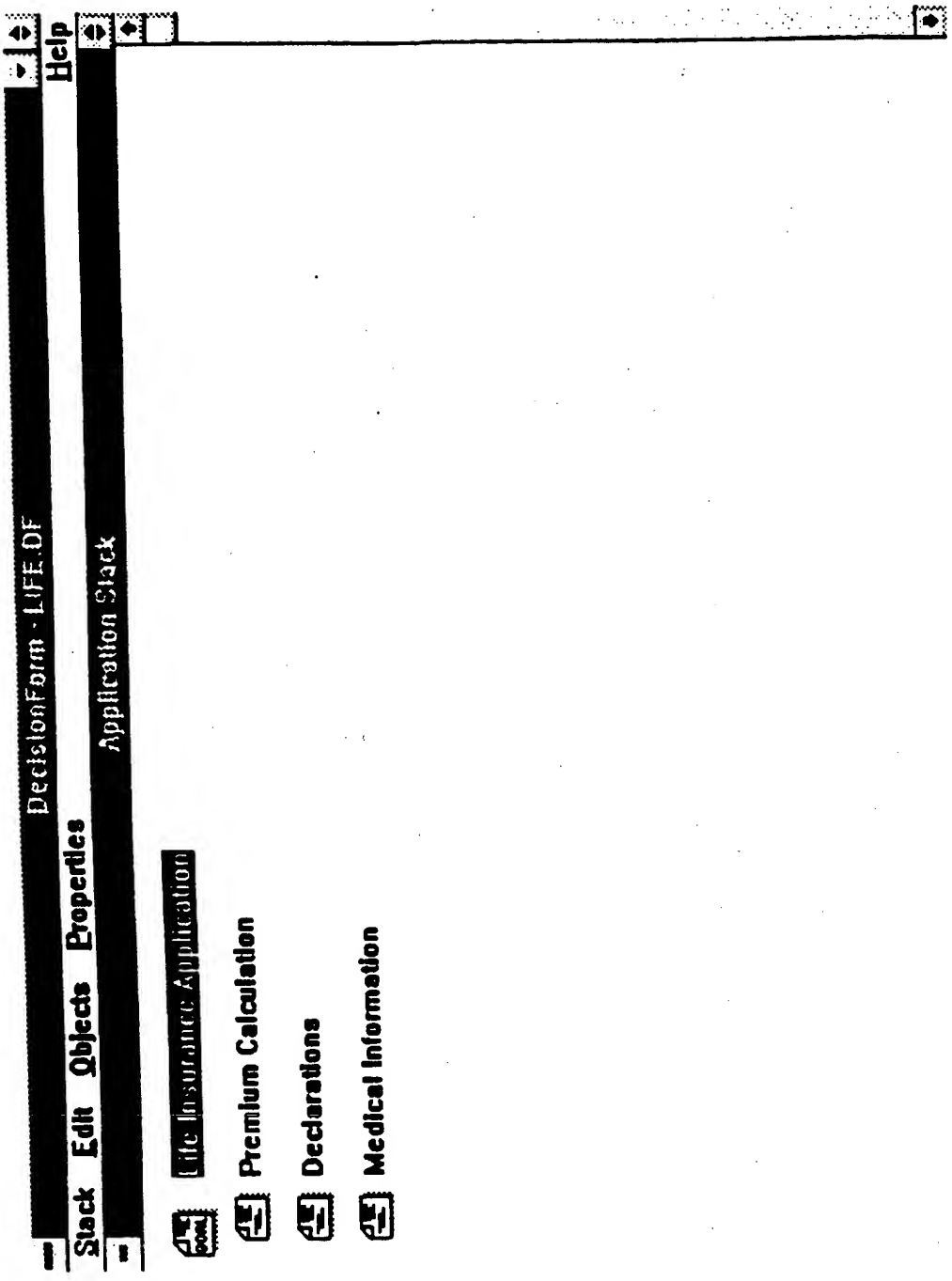
DecisionForm - LIFE.DF
Help
Stack Edit Objects Properties
Application Stack
Life Insurance Application
Premium Calculation
Declarations
Medical Information

Fig. 23

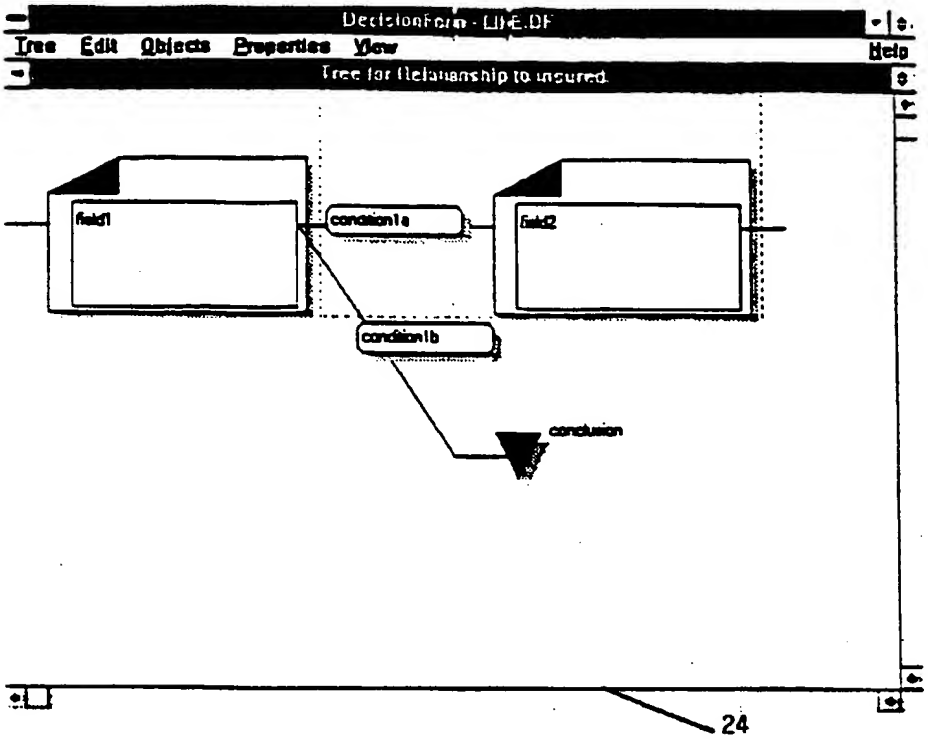
Tree Edit Objects Properties View
Help
field1
condition1a
field2
condition1b
conclusion
24

Fig. 24

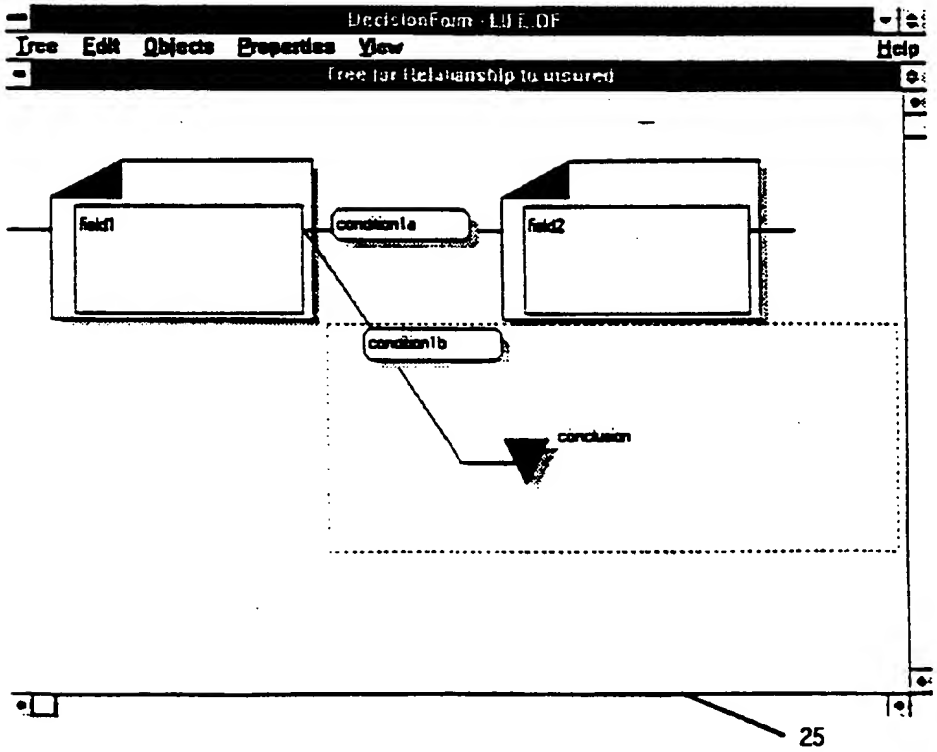
Tree Edit Objects Properties View
Help
field1
condition1a
field2
condition1b
conclusion
25

Fig. 25

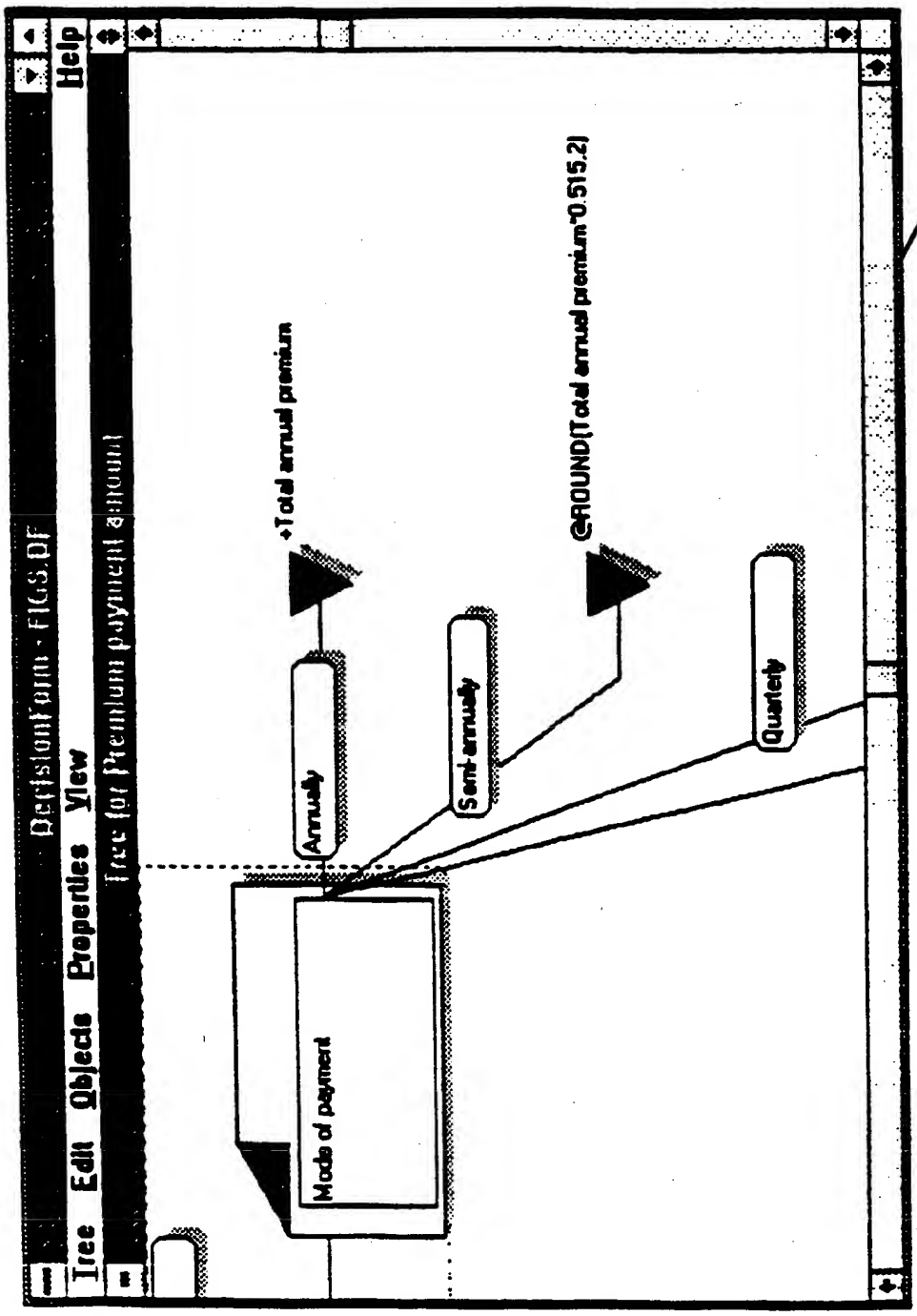
Help
View
Properties
Objects
Edit
Tree
Mode of payment
Annuity
Semi-annuity
Quarterly
+Total annual premium
@ROUND(Total annual premium*0.515,2)
26

Fig. 26

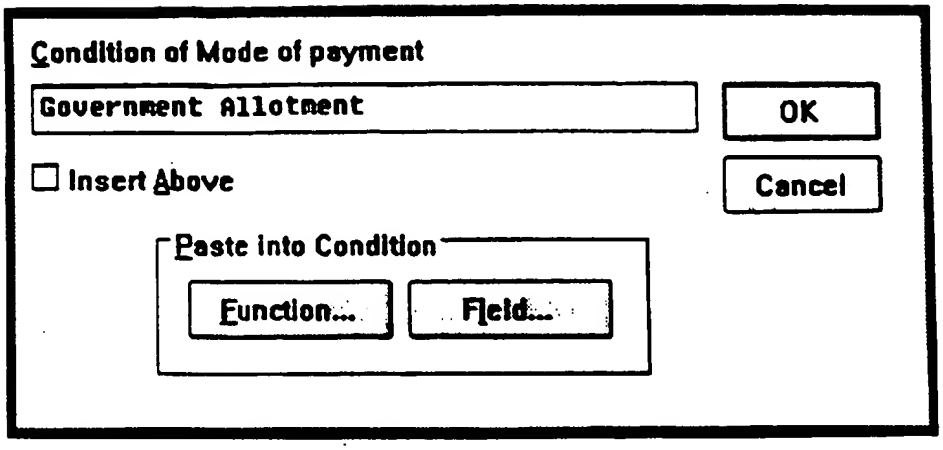

Fig. 27

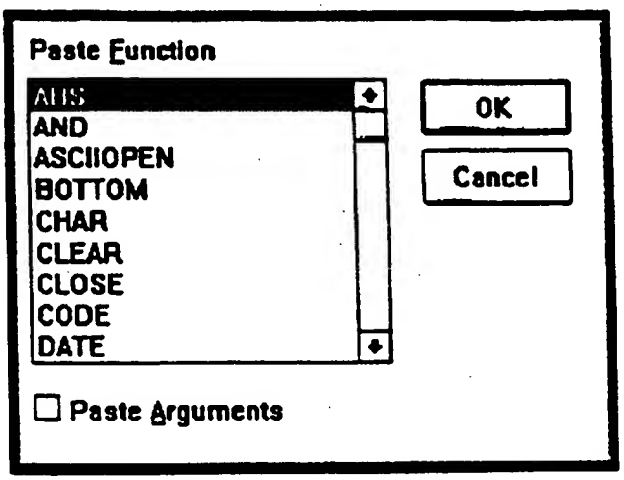

Fig. 28

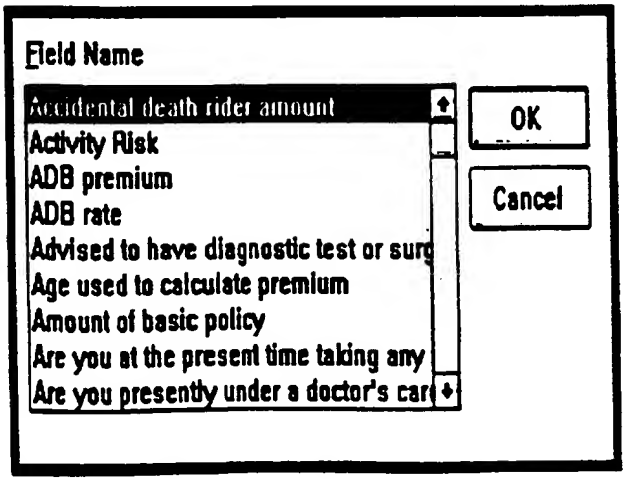

Fig. 29

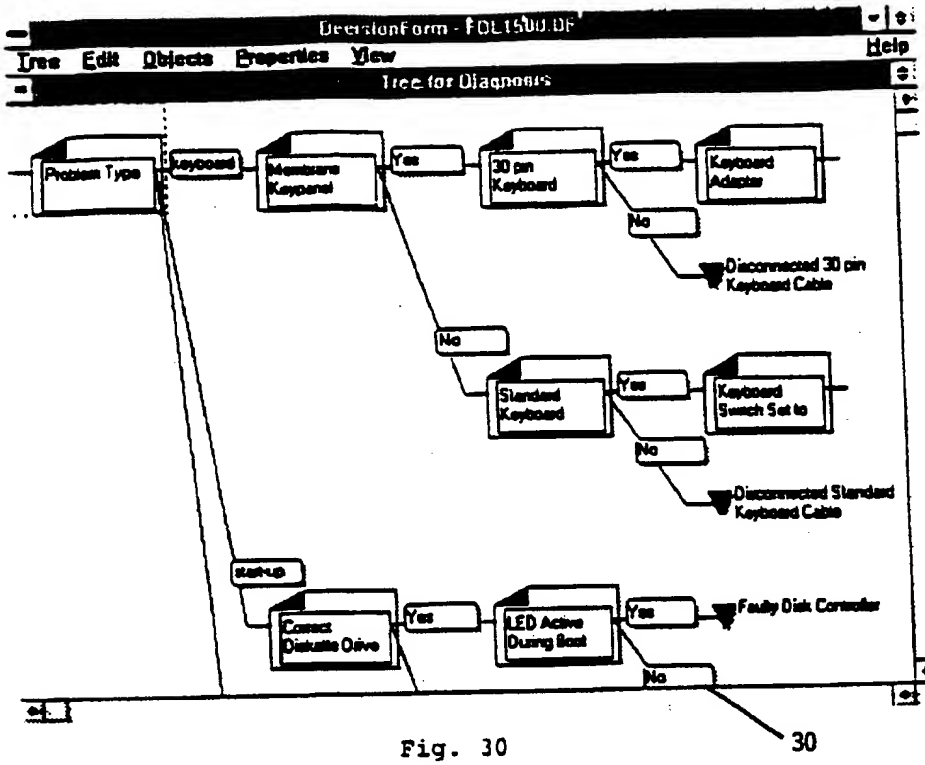
DecisionForm - FDL1500.DF
Tree Edit Objects Properties View Help
Tree for Diagnosis
Problem Type
Keyboard
Membrane Keyboard
Yes
30 pin Keyboard
Yes
Keyboard Adapter
No
Disconnected 30 pin Keyboard Cable
No
Standard Keyboard
Yes
Keyboard Switch Set to
No
Disconnected Standard Keyboard Cable
start-up
Correct Diskette Drive
Yes
LED Active During Boot
Yes
Faulty Disk Controller
No
30

Fig. 30

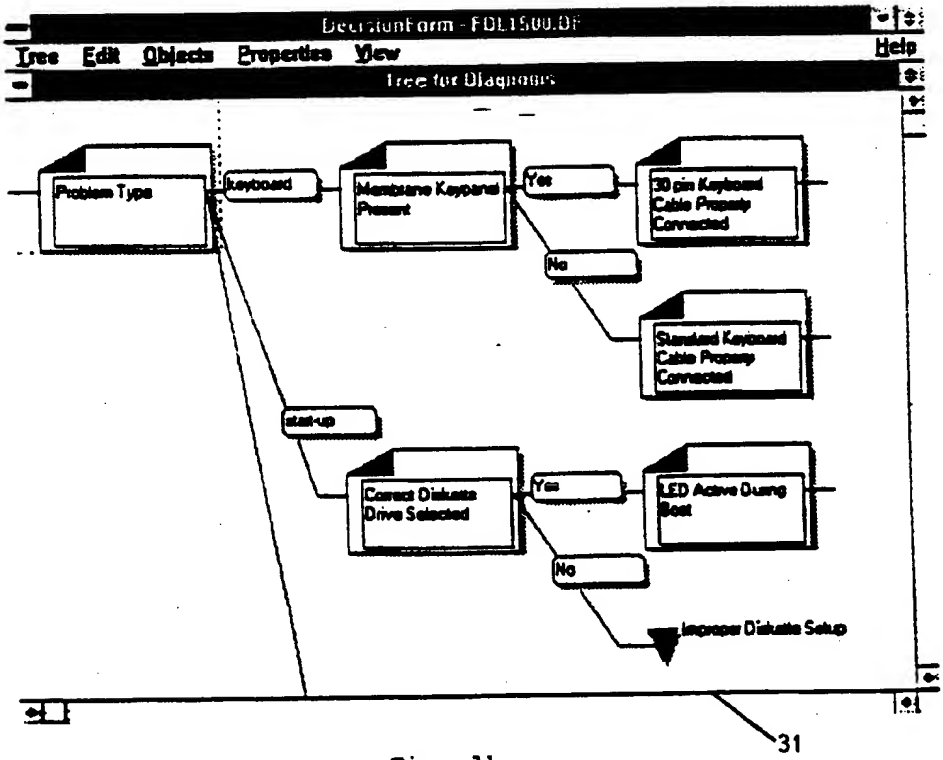
DecisionForm - FDL1500.DF
Tree Edit Objects Properties View Help
Tree for Diagnosis
Problem Type
Keyboard
Membrane Keyboard Present
Yes
30 pin Keyboard Cable Properly Connected
No
Standard Keyboard Cable Properly Connected
start-up
Correct Diskette Drive Selected
Yes
LED Active During Boot
No
Improper Diskette Setup
31

Fig. 31

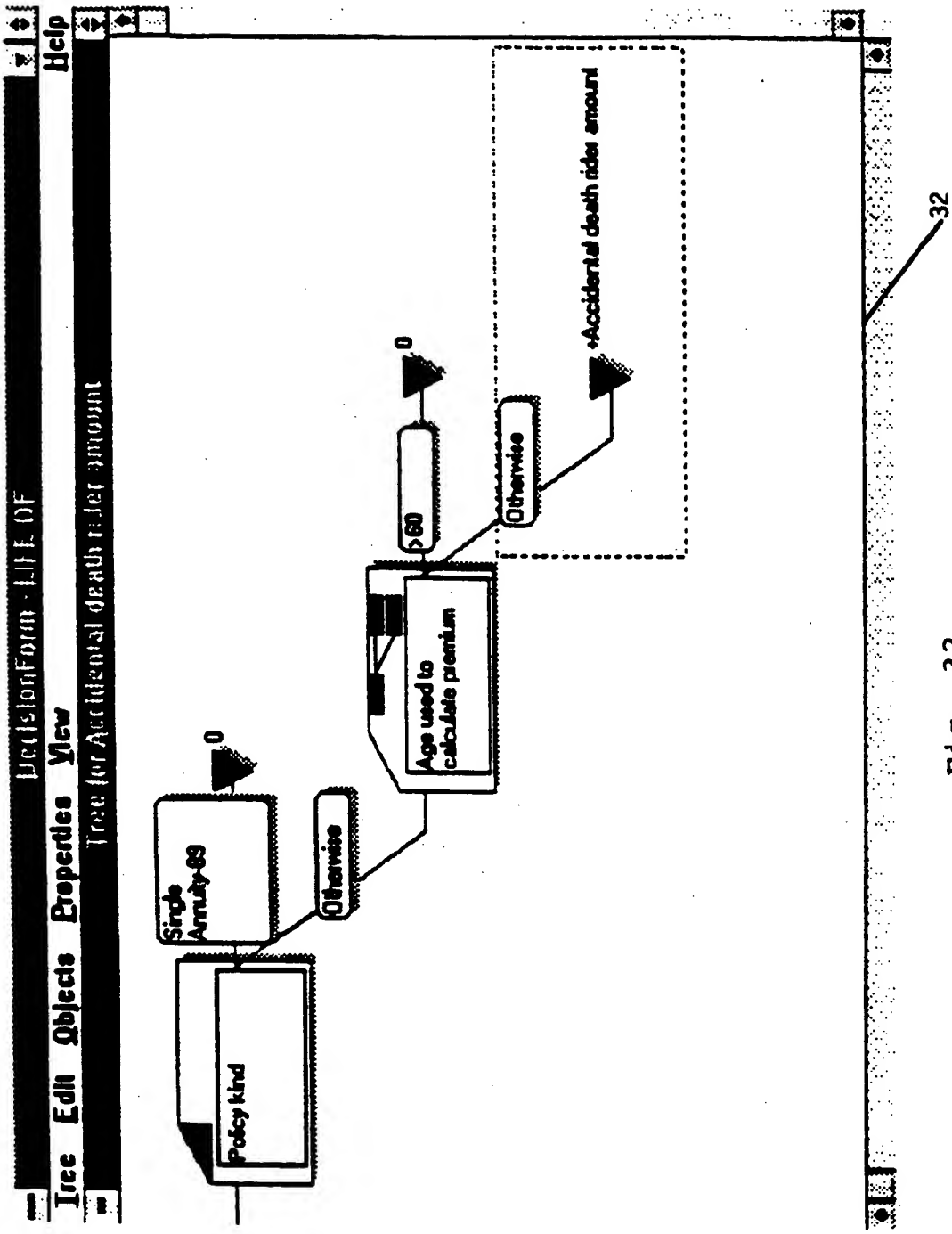
DecisionForum - LIFE OF
Help
Tree Edit Objects Properties View
Tree for Accidental death rider amount
Policy kind
Single Annuity-83
0
Otherwise
Age used to calculate premium
>60
0
Otherwise
Accidental death rider amount
32

Fig. 32

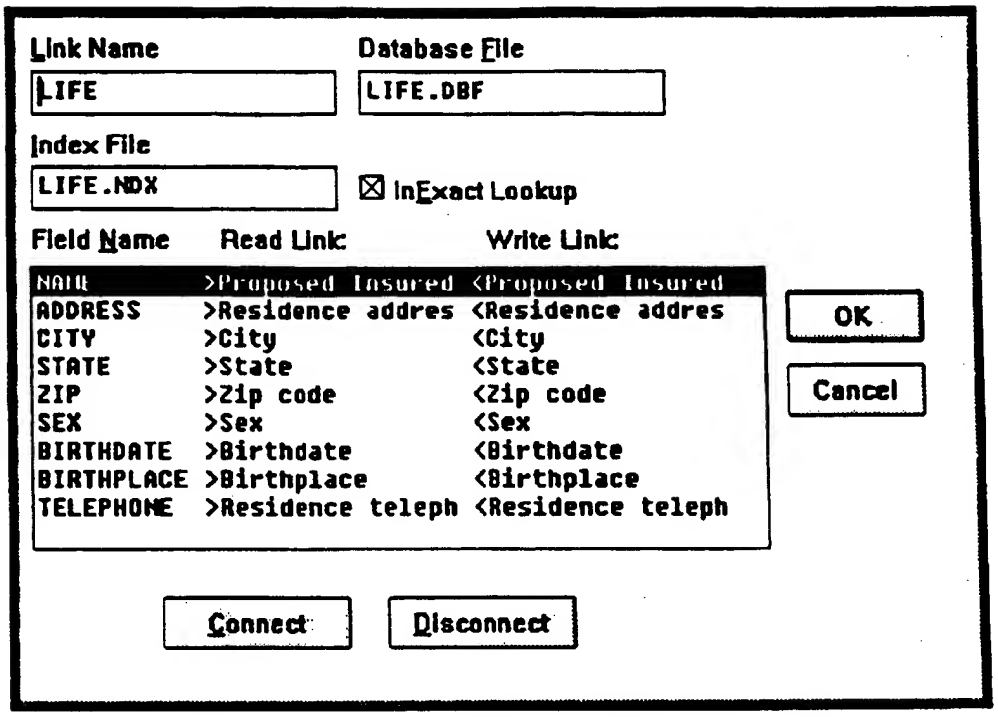
Link Name
LIFE
Database File
LIFE.DBF
Index File
LIFE.NDX
InExact Lookup
Field Name
Read Link:
Write Link:
NAME >Proposed Insured <Proposed Insured
ADDRESS >Residence addres <Residence addres
CITY >City <City
STATE >State <State
ZIP >Zip code <Zip code
SEX >Sex <Sex
BIRTHDATE >Birthdate <Birthdate
BIRTHPLACE >Birthplace <Birthplace
TELEPHONE >Residence teleph <Residence teleph
OK
Cancel
Connect
Disconnect

Fig. 33

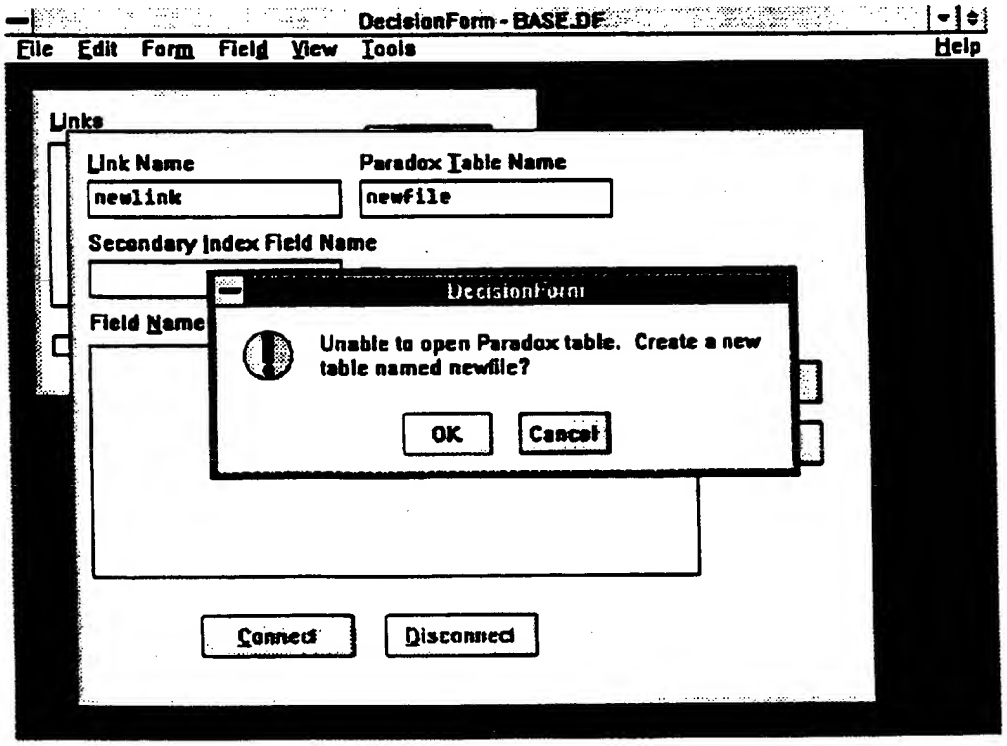
DecisionForm - BASE.DF
File Edit Form Field View Tools Help
Links
Link Name
newlink
Paradox Table Name
newfile
Secondary Index Field Name
Field Name
DecisionForm
Unable to open Paradox table. Create a new table named newfile?
OK
Cancel
Connect
Disconnect

Fig. 34